Navigating the Alpha Jungle: An LLM-Powered MCTS Framework for Formulaic Alpha Factor Mining

Yu Shi[1], **Yitong Duan**[1, 2], **and Jian Li**[1]

[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Zhongguancun Institute of Artificial Intelligence
{shi-y23, dyt19}@mails.tsinghua.edu.cn, lapordge@gmail.com

Abstract

Alpha factor mining is pivotal in quantitative investment for identifying predictive signals from complex financial data. While traditional formulaic alpha mining relies on human expertise, contemporary automated methods, such as those based on genetic programming or reinforcement learning, often struggle with search inefficiency or yield alpha factors that are difficult to interpret. This paper introduces a novel framework that integrates Large Language Models (LLMs) with Monte Carlo Tree Search (MCTS) to overcome these limitations. Our framework leverages the LLM's instruction-following and reasoning capability to iteratively generate and refine symbolic alpha formulas within an MCTS-driven exploration. A key innovation is the guidance of MCTS exploration by rich, quantitative feedback from financial backtesting of each candidate factor, enabling efficient navigation of the vast search space. Furthermore, a frequent subtree avoidance mechanism is introduced to enhance search diversity and prevent formulaic homogenization, further improving performance. Experimental results on real-world stock market data demonstrate that our LLM-based framework outperforms existing methods by mining alphas with superior predictive accuracy and trading performance. The resulting formulas are also more amenable to human interpretation, establishing a more effective and efficient paradigm for formulaic alpha mining.

Extended version — https://arxiv.org/abs/2505.11122

Introduction

Predicting price movements in financial markets, characterized by low signal-to-noise ratios, remains a central challenge in quantitative investment. A common strategy to enhance model predictiveness is the extraction of predictive signals, or alpha factors (hereafter simply "alphas"), from stock data (Qian, Hua, and Sorensen 2007; Tulchinsky 2019). Current alpha factor mining methodologies broadly fall into two categories: neural network-based and formula-based. Neural approaches (e.g., FactorVAE (Duan et al. 2022), HIST (Xu et al. 2021a), REST (Xu et al. 2021b)) implicitly construct complex alphas via deep learning, capturing intricate patterns but often suffering from a lack of interpretability. In contrast, formula-based methods aim to discover alphas represented by explicit mathematical expressions. These alpha factors are traditionally human-crafted, reflecting market insights (e.g., Fama-French factors (Fama and French 1992), financial anomalies (Harvey, Liu, and Zhu 2016; Hou, Xue, and Zhang 2020)). In recent years, automated techniques have emerged, employing methods like genetic programming or reinforcement learning to discover such formulaic alphas (Shi et al. 2024; Yu et al. 2023; Zhang et al. 2020, 2023).

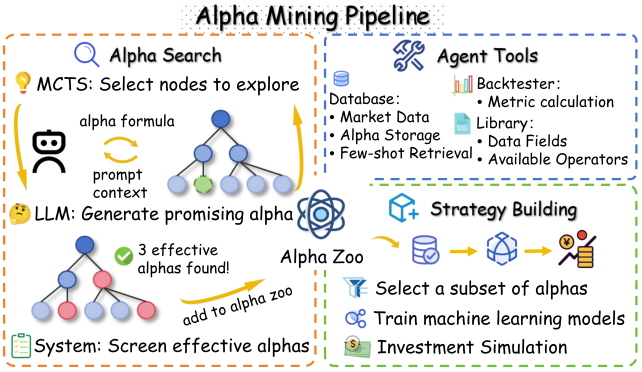

Figure 1: A high-level schematic of our proposed alpha mining pipeline. The pipeline features an iterative Alpha Search loop where an LLM, guided by MCTS, generates and refines formulas. Effective alphas are collected in an Alpha Zoo before being used in the final Strategy Building stage.

Despite their promise, existing automated formulaic alpha mining approaches face significant limitations. **First, the discovered alphas often exhibit poor interpretability.** These automated methods frequently engage in unconstrained, data-driven exploration of the vast alpha space, often without sufficient guidance from financial theory or domain expertise. Consequently, the resulting formulas can be overly complex and opaque. This lack of transparency poses considerable challenges in practical investment scenarios: it hinders practitioners' ability to understand the underlying economic rationale of a strategy, makes it difficult to attribute portfolio performance accurately, and can erode trust, thereby impeding the adoption of these alphas even if they show promise in backtests. **Second, current methodologies often suffer from search inefficiency.** The search for a suf-

ficient number of effective alpha factors typically requires generating and evaluating an enormous volume of candidate formulas. This exhaustive search process, while necessary due to the low signal density, inherently increases the likelihood of discovering spurious relationships and overfitting to the training data (Harvey, Liu, and Zhu 2016). As a result, many discovered alphas may exhibit poor generalization and deliver underwhelming out-of-sample performance.

Addressing the identified shortcomings necessitates innovative approaches. In this light, Large Language Models (LLMs) emerge as a promising direction, given their vast prior knowledge and strong reasoning capabilities which are well-suited for generating interpretable alphas—a potential demonstrated in analogous tasks like financial investment (Yu et al. 2024) and code generation (Li et al. 2024a). Drawing inspiration from advancements in LLM reasoning (e.g., Chain-of-Thought (Wei et al. 2022), Tree-of-Thought (Yao et al. 2024)) and the efficacy of Monte Carlo Tree Search (MCTS) (Coulom 2007; Silver et al. 2016) in enhancing LLM performance on complex problems (Zhang et al. 2024), we frame alpha mining as an MCTS-driven search problem. Within this framework, each node in the tree represents a candidate alpha formula, allowing for a systematic exploration and iterative refinement within the vast and complex alpha space. Figure 1 provides a high-level illustration of this entire pipeline.

Unlike tasks such as mathematical derivation, where evaluating the contribution of intermediate steps towards the final solution is often challenging before the derivation is complete, alpha factor mining provides fine-grained feedback on each candidate alpha formula through backtesting. We leverage this detailed feedback to guide our search. We initiate the search with a LLM-generated alpha formula, as the root node of the search tree. Then we utilize the LLM to iteratively refine and improve the formulas, expanding the tree with new, potentially superior nodes. Furthermore, to mitigate the homogeneity of generated alpha formulas, we conduct frequent subtree mining on effective alphas and explicitly instruct the LLM to avoid using the most frequent subtrees during generation. By explicitly diversifying the search away from these common motifs, our approach enhances search efficiency and improves the quality of discovered alphas.

The synergy between MCTS and LLMs has indeed shown promise in various reasoning tasks (Zhao, Lee, and Hsu 2023; DeLorenzo et al. 2024). Our work is distinct in its specific application of this synergy to the unique challenges of formulaic alpha mining. Unlike general reasoning tasks that use MCTS to explore a set of predefined actions or have the LLM evaluate abstract states (Xie et al. 2024; Li et al. 2025; Dainese et al. 2024), our framework leverages the LLM as a *generative prior* for symbolic alpha formulas. Crucially, the MCTS exploration is guided by rich, quantitative, and domain-specific feedback from financial backtesting performed on each candidate alpha. This iterative loop—where the LLM's generative capabilities are steered by MCTS informed by empirical financial performance—offers a distinct advantage.

Our main contributions can be summarized as follows:

- We propose an LLM-Powered MCTS framework for formulaic alpha mining, modeling the task as a tree search-based reasoning problem where the LLM performs multi-step formula refinement guided by detailed backtesting feedback.
- We design a frequent subtree avoidance method to improve search efficiency and alpha effectiveness by guiding the LLM to explore less common yet potentially effective formula structures.
- We conduct a series of experiments to demonstrate the effectiveness of our proposed framework. The alphas mined by our framework achieve superior prediction performance while maintaining good interpretability, compared to those from other methods.

Preliminary

Alpha Factor Mining

We consider a financial market with n stocks observed over T trading days. For each stock $i \in \{1, \dots, n\}$ and day $t \in \{1, \dots, T\}$, its state is described by a feature vector $\boldsymbol{x}_{i,t} \in \mathbb{R}^m$. Raw features include daily open, high, low, close prices (OHLC), trading volume, and Volume-Weighted Average Price (VWAP). The complete market history is a tensor $\boldsymbol{X} \in \mathbb{R}^{T \times n \times m}$. Correspondingly, future returns are organized in a matrix $\boldsymbol{Y} \in \mathbb{R}^{T \times n}$, where $y_{i,t}$ is the realized future return for stock i subsequent to day t. To capture temporal patterns, we use a lookback window of length τ. An *alpha factor*, f, maps the historical feature data for this window, $\boldsymbol{X}_{t-\tau+1:t} = \{\boldsymbol{X}_s \mid t - \tau < s \leq t\}$, to a vector of predictive scores $\boldsymbol{v}_t = f(\boldsymbol{X}_{t-\tau+1:t}) \in \mathbb{R}^n$. Each $v_{i,t}$ represents the alpha's assessment of stock i's future return.

Alpha factor mining aims to discover a diverse set of K alphas, $\mathcal{F} = \{f_1, \dots, f_K\}$, by searching within the vast space of all possible alpha factors, denoted as $\mathcal{A}$. The outputs of these individual alphas, $\{\boldsymbol{v}_{k,t} = f_k(\boldsymbol{X}_{t-\tau+1:t})\}_{k=1}^K$, are typically aggregated by a combination model, g, into a composite alpha vector $\boldsymbol{z}_t = g(\{\boldsymbol{v}_{k,t}\}_{k=1}^K; \boldsymbol{\theta}_g)$, where $\boldsymbol{\theta}_g$ are model parameters. The quality of this composite signal is evaluated using a predefined performance metric, $\mathcal{L}$ (e.g., Information Coefficient). Optimal parameters for the combination model, $\boldsymbol{\theta}_g^*$, are learned by maximizing this metric:

$$\boldsymbol{\theta}_g^*(\mathcal{F}) = \arg\max_{\boldsymbol{\theta}_g} \mathcal{L}(g(\{\boldsymbol{v}_{k,t}\}_{k=1}^K; \boldsymbol{\theta}_g), \boldsymbol{Y}). \quad (1)$$

Let $g^*(\mathcal{F})$ be the combination model with parameters $\boldsymbol{\theta}_g^*(\mathcal{F})$. The overarching goal is to find an optimal set of alpha factors $\mathcal{F}^* \subset \mathcal{A}$ that maximizes the performance of this optimally combined signal: $\mathcal{F}^* = \arg\max_{\mathcal{F} \subset \mathcal{A}} \mathcal{L}(g^*(\mathcal{F}), \boldsymbol{Y})$. This constitutes a challenging bilevel optimization problem due to the immense search space $\mathcal{A}$ and the complex interactions between alpha factors.

Formulaic Alpha

In this work, we focus on *formulaic alphas*: alpha factors defined by mathematical expressions. These expressions are constructed from operators and operands. Operands typically include raw input features (e.g., $\text{close}_{i,t}$) and numerical constants. Operators apply mathematical transforma-

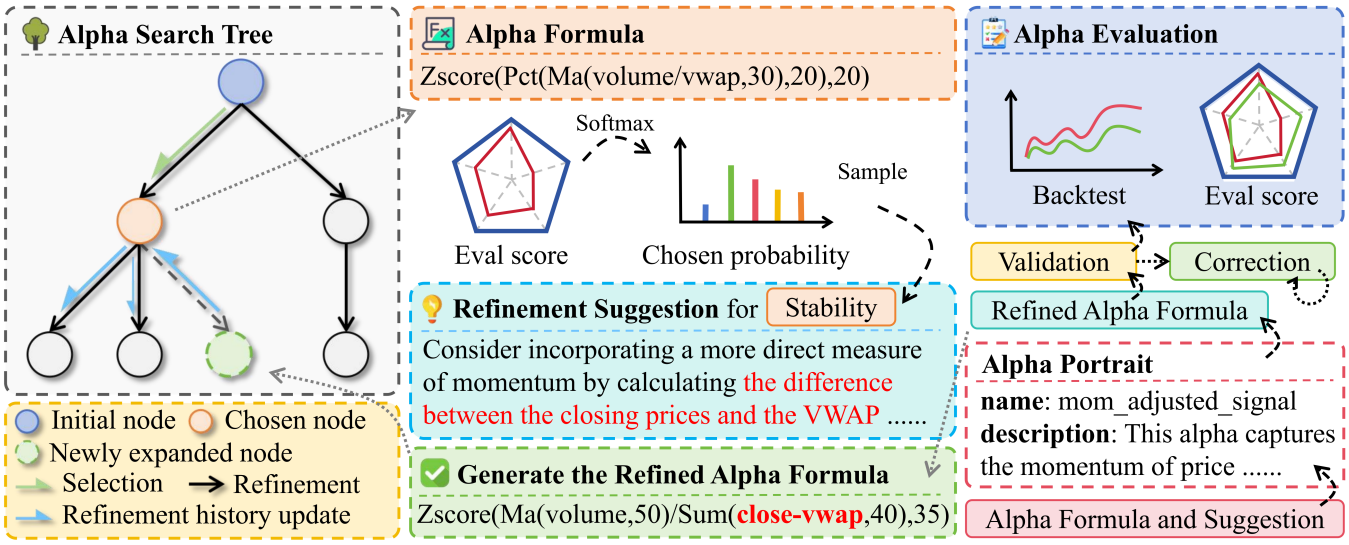

Figure 2: Overview of our LLM-powered MCTS framework. The process begins with node selection via UCT. A refinement dimension is then chosen based on the node's multi-dimensional evaluation scores. The LLM first proposes a conceptual refinement suggestion for that dimension and then translates it into a concrete formula. The new formula is backtested, and its performance results are used to expand the tree with a new node.

tions; for example, time-series operators can be used to construct an alpha like $\text{Ma}(\text{close}, 5) - \text{Ma}(\text{close}, 20)$. This specific alpha captures a price trend by contrasting short-term with long-term moving averages of closing prices. Formulaic alphas are naturally represented as expression trees (leaf nodes: operands; internal nodes: operators), making their structured yet flexible nature amenable to the automated mining techniques central to our framework.

Methodology

Our proposed alpha mining framework integrates LLMs with MCTS to automate the discovery and refinement of alpha factors. The framework's objective is to search within the vast space of possible alpha formulas, denoted as $\mathcal{A}$, to find a set of high-performing alphas. Figure 2 provides a conceptual illustration. The core iterative process involves: (1) selecting a promising node (alpha formula) using the Upper Confidence Bound for Trees (UCT) criterion (Kocsis and Szepesvári 2006); (2) expanding this node by having the LLM generate a refined alpha, guided by performance feedback on specific evaluation dimensions; and (3) evaluating the new alpha via backtesting, with results forming a new node in the search tree $\mathcal{T}$. The LLM's role is twofold: first, to propose targeted refinement suggestions, and second, to translate these suggestions into a concrete alpha formula $f \in \mathcal{A}$. Iteratively, high-performing alphas that satisfy a set of predefined criteria $\mathcal{C}$ (e.g., IC > 0.02) are collected into an effective alpha repository, $\mathcal{F}_{zoo}$.

Selection

The selection step in our MCTS framework navigates the exploration-exploitation trade-off. Each node $s \in \mathcal{T}$ represents an alpha, characterized by its formula f_s and refinement history. An action a corresponds to a specific refinement applied to s. Each state-action pair (s, a) maintains a quality value $Q(s, a)$, representing the maximum reward (alpha score) observed in the subtree rooted at the child node resulting from this action. We employ the UCT criterion to select the optimal action a^*:

$$a^* = \arg\max_{a \in A(s)} \left(Q(s,a) + c\sqrt{\frac{\ln(N_s)}{N_{s'}}} \right) \tag{2}$$

where $A(s)$ is the set of existing actions from state s, N_s is the visit count of the parent state s, $N_{s'}$ is the visit count of the child state $s' = \text{child}(s, a)$, and c is the exploration weight.

Unlike standard MCTS, which expands only leaf nodes, our approach allows any node to be selected for expansion. This is crucial for iteratively refining promising, but not terminal, alpha ideas. To enable this, we augment the action space $A(s)$ at any internal node s with a "virtual" expansion action, a_e. The selection process thus considers the full action set $A(s) \cup \{a_e\}$. The UCT score for this virtual action is computed by adapting Equation 2, where we define a virtual visit count for the prospective new node s'_e as $N_{s'_e} = 1 + |C(s)|$, with $C(s)$ being the set of existing children of s. If a_e is selected, node s is chosen for expansion. This mechanism ensures that promising, non-leaf nodes can be further refined.

Expansion

Upon selecting a node s for expansion, a new, refined alpha factor f_{new} is generated. This process is structured to enhance the LLM's effectiveness and the quality of refinements.

Dimension-Targeted Refinement Suggestion. Each node s is associated with a multi-dimensional evaluation score vector $\boldsymbol{E}_s = [e_1, \ldots, e_q] \in [0, e_{\max}]^q$. To guide refinement towards areas of weakness while maintaining exploratory diversity, we stochastically select a target dimension i^* for improvement. The probability of choosing dimension i is defined as:

$$P(i^* = i|s) = \text{Softmax}\left((e_{\max} \cdot \mathbf{1}_q - \boldsymbol{E}_s)/T\right)_i \quad (3)$$

where $\mathbf{1}_q$ is a q-dimensional vector of ones and T is a temperature parameter. This strategy prioritizes dimensions with lower scores. Once a dimension i^* is selected, the LLM generates a textual refinement suggestion d_{s,i^*} aimed at improving performance on that dimension. This is framed as a few-shot learning task, where the context includes effective alphas from $\mathcal{F}_{zoo}$.

Alpha Formula Generation and Validation. Following the targeted suggestion, we employ a two-step generation process. First, the LLM articulates the refined conceptual hypothesis, which is then used to prompt the generation of the concrete formula. This process, ensuring the formula aligns with a clear investment rationale, can be formalized as:

$$d_{s,i^*} \sim p_{\text{LLM}}(\cdot|s, i^*, \mathcal{F}_{zoo}) \quad (4)$$

$$f_{new} \sim p_{\text{LLM}}(\cdot|d_{s,i^*}, f_s) \quad (5)$$

The generated formula f_{new} undergoes an automated validation check, $\text{IsValid}(f_{new})$. If invalid, feedback is provided to the LLM for iterative correction. A valid formula and its evaluation results constitute the new node s_{new} in the MCTS tree.

Multi-Dimensional Alpha Evaluation

The evaluation of a candidate alpha f is performed directly via backtesting, bypassing the simulation phase of traditional MCTS. A key challenge is the evolving nature of the effective alpha repository $\mathcal{F}_{zoo}$, which progressively raises the bar for new alphas. To address this, we employ a relative ranking approach. The rank of f against the repository for a given metric m is:

$$R(f, m, \mathcal{F}_{zoo}) = \frac{1}{|\mathcal{F}_{zoo}|} \sum_{f' \in \mathcal{F}_{zoo}} \mathbb{I}(m(f) < m(f')) \quad (6)$$

where $\mathbb{I}(\cdot)$ is the indicator function. This provides an adaptive evaluation criterion, avoiding fixed thresholds that may be too stringent early on or too lenient later.

To provide granular feedback for refinement, we evaluate f across a set of q dimensions $\mathcal{D}$, exemplified here by **Effectiveness**, **Stability**, **Turnover**, **Diversity**, and **Overfitting Risk**. For each dimension $D_i \in \mathcal{D} \setminus \{\text{Overfitting Risk}\}$, we compute a score e_i based on its percentile rank using a corresponding metric m_i:

$$e_i(f) = 1 - R(f, m_i, \mathcal{F}_{zoo}) \quad (7)$$

The assessment of **Overfitting Risk**, denoted e_{overfit}, is distinct. We leverage an LLM that analyzes the formula f and its refinement history $H(s)$, providing a qualitative judgment: $e_{\text{overfit}} = \text{LLM}_{\text{eval}}(f, H(s))$. The overall alpha score,

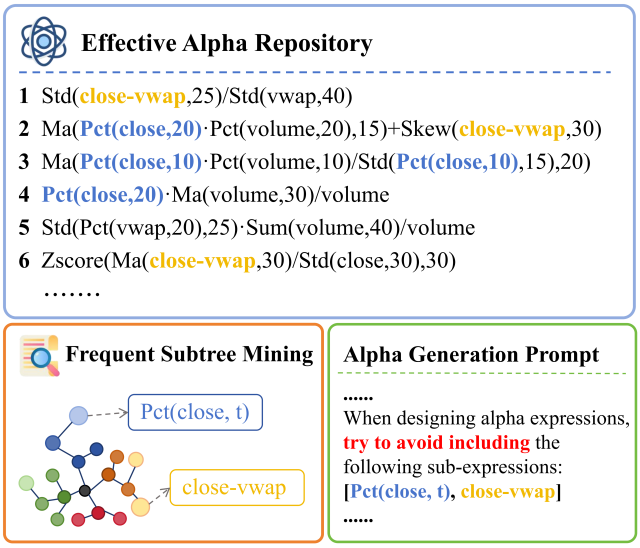

Figure 3: Illustration of the Frequent Subtree Avoidance (FSA). The set of effective alphas from the Alpha Repository is mined for frequent subtrees. The most frequent ones are identified, and the LLM is subsequently instructed to avoid generating new formulas containing these common structural motifs.

which serves as the reward signal for MCTS, is the aggregate of these dimensional scores:

$$S(f) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{q} e_i(f) \quad (8)$$

This score $S(f)$ is used to update the Q-values in the search tree.

Backpropagation

During backpropagation, the reward $S(f_{new})$ of the newly evaluated alpha at node s_{new} updates the statistics of all nodes along the path from the root to its parent. For each ancestor node s_k on this path, leading to its child s_{k+1} via action a_k, we perform the following updates:

$$N_{s_k} \leftarrow N_{s_k} + 1 \quad (9)$$

$$Q(s_k, a_k) \leftarrow \max(Q(s_k, a_k), S(f_{new})) \quad (10)$$

This ensures that the value of a path reflects the best outcome discovered within its entire subtree. Crucially, to enhance the quality of subsequent refinement suggestions and the accuracy of overfitting assessment, the LLM is provided with rich contextual information: the refinement history of the current node's parent, children, and siblings. This allows the LLM to analyze the refinement trajectory and avoid redundant suggestions.

Frequent Subtree Avoidance

To mitigate alpha formula homogenization and prevent the over-exploitation of common motifs, we introduce Frequent Subtree Avoidance (FSA), inspired by the concept of "root

genes" in AutoAlpha (Zhang et al. 2020). We define a *root gene* as a subtree in an alpha's expression tree whose leaves are exclusively raw input features (e.g., 'close', 'high'). To focus on structure, we use an operator $\text{Abs}(\cdot)$ that abstracts away concrete parameter values from an expression tree. For instance, $\text{Abs}(\text{Ma}(\text{vwap}, 20))$ becomes $\text{Ma}(\text{vwap}, t)$. The set of abstracted root genes for an alpha f is denoted $\bar{\mathcal{G}}(f)$.

The FSA mechanism operates by first identifying frequent closed root genes from the repository $\mathcal{F}_{zoo}$. A root gene is "closed" if none of its immediate supertrees share the same support count, which helps identify maximal common patterns. The support for an abstracted root gene $\bar{g}$ is:

$$\text{Support}(\bar{g}) = \frac{1}{|\mathcal{F}_{zoo}|} \sum_{f' \in \mathcal{F}_{zoo}} \mathbb{I}(\bar{g} \subseteq \bar{\mathcal{G}}(f')) \qquad (11)$$

We select the top-k most frequent closed root genes to form a set of forbidden structures, $\mathcal{G}_{\text{forbidden}}$. During generation (Eq. 5), the LLM is constrained to produce a new formula f_{new} that avoids these motifs:

$$\text{Constraint: } \bar{\mathcal{G}}(f_{new}) \cap \mathcal{G}_{\text{forbidden}} = \emptyset \qquad (12)$$

FSA acts as a regularization on the generation process. By discouraging common structures, it guides the MCTS search towards more structurally diverse and potentially novel regions of the alpha space $\mathcal{A}$, enabling a more efficient exploration, as demonstrated in Section .

Experiment

We evaluate our proposed framework on real-world stock market data, addressing the following research questions (RQs):

Q1: How does our approach compare to baselines in predictive performance?

Q2: Are MCTS and Frequent Subtree Avoidance effective components within our framework?

Q3: How does the interpretability of alpha formulas mined by our method compare to others?

Experiment Settings

Data Our experiments are conducted on the Chinese A-shares market. To ensure comprehensive market representation, our experiments separately target two stock pools: the CSI 300 Index (large-cap, liquid stocks) and the CSI 1000 Index (small to mid-cap stocks). We define two distinct prediction targets: the 10-day return and the 30-day return of the stocks, with buying and selling executed at the closing price. The dataset is split chronologically into training (2011/01/01–2020/12/31) and testing (2021/01/01–2024/11/30) periods.

Baselines for Comparison We compare our framework with several formulaic alpha mining methods. **DSO** (Deep Symbolic Optimization) (Landajuela et al. 2022) is a deep learning framework for symbolic optimization. **GP** employs genetic programming for alpha mining. **AlphaGen** (Yu et al. 2023) is a reinforcement learning framework for formulaic alpha mining. **AlphaForge** (Shi et al. 2024) features a generative-predictive architecture; to ensure a fair comparison of generative capabilities, we use only its alpha mining network. Among LLM-based approaches, **CoT** (Chain-of-Thought) (Wei et al. 2022) prompts LLMs for step-by-step reasoning to directly generate alpha factors. **ToT** (Tree-of-Thought) (Yao et al. 2024) enables LLMs to explore diverse reasoning paths in a tree structure. Lastly, **FAMA** (Li et al. 2024b) leverages LLMs with in-context examples to diversify formulas and a "chain-of-experience" to learn from past successes.

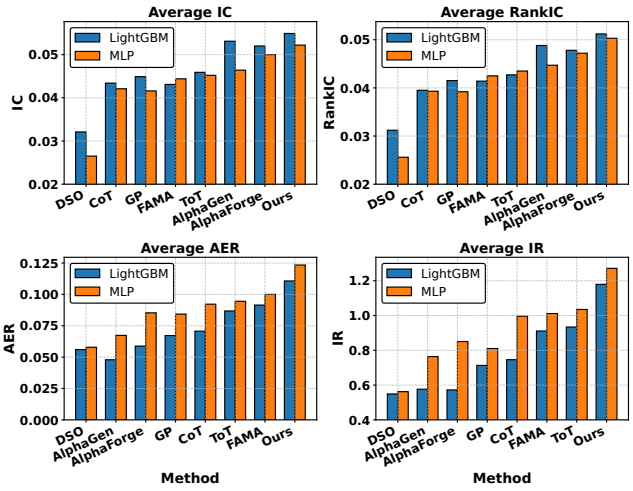

Figure 4: The average performance of LightGBM and MLP models trained on alphas mined by different methods.

We use OpenAI's GPT4.1 model as the LLM for both our method and the LLM-based baselines. To ensure a fair and rigorous comparison of algorithmic efficiency, we benchmark all methods based on a controlled "search count" (i.e., the number of unique alpha formulas generated and evaluated). This metric normalizes for the vast differences in computational cost per generation step across methods (e.g., a single LLM call vs. a GP mutation) and provides a direct measure of search space exploration efficiency. This approach is well-suited because our framework and all baselines inherently involve a distinct search process, where each iteration yields a new candidate alpha formula. For LLM-based methods, we report the best performance achieved with search counts of 1,000, 2,000, or 3,000. For other methods, the search count is incrementally increased from a small value until performance converges, capped at 600,000 (200× the LLM-based methods' maximum). This experimental design facilitates two key comparisons: first, it allows for an equitable assessment of our framework against other LLM-based methods under similar, well-defined computational budgets; second, it enables a robust comparison of search efficiency against other non-LLM-based methods.

Experiment 1: Prediction Performance Comparison

We evaluate the predictive performance of alphas generated by our method against baselines. To comprehensively assess the effectiveness of the generated alpha sets, we employ two representative machine learning models: Light-

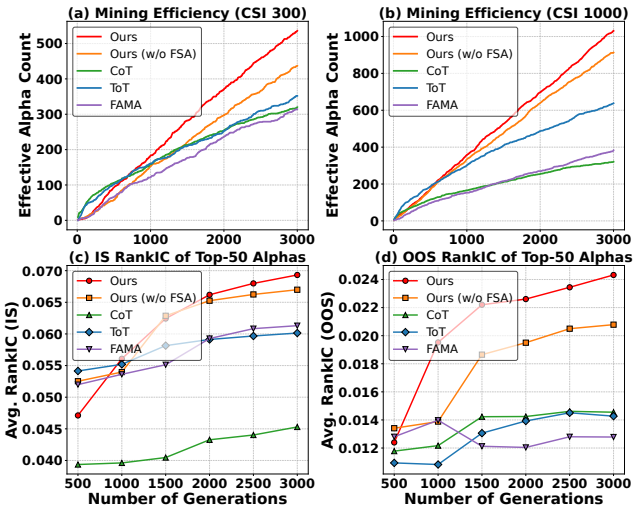

Figure 5: Analysis of search dynamics. (a, b) Alpha mining efficiency, measured as the count of effective alphas found versus total generated. (c, d) Average in-sample and out-of-sample RankIC of the top 50 alphas over generations.

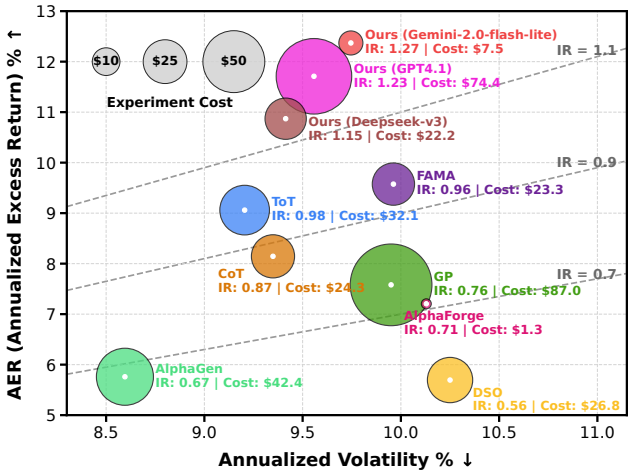

Figure 6: Cost-performance analysis across various methods. The plot shows Annualized Excess Return (AER) vs. Annualized Volatility (lower is better). Bubble size indicates the estimated single-run cost. Dashed lines represent constant Information Ratio (IR) levels.

GBM (Ke et al. 2017), a popular gradient boosting framework known for its efficiency, and a 3-layer Multi-Layer Perceptron (MLP), which can capture complex non-linear relationships. For each alpha generation method, we create alpha sets of three distinct sizes—10, 50, and 100—to serve as input features for these models. This allows for a thorough comparison of the mined alpha sets across varying sizes. Both input alphas and target returns undergo cross-sectional rank normalization before training to mitigate outlier influence. The predictive power of the alphas is evaluated using two standard metrics in finance: the Information Coefficient (IC) and the Rank Information Coefficient (RankIC).

To assess the practical profitability of the mined alphas in simulated real-world stock market scenarios, we follow established evaluation methodologies (Yu et al. 2023) and conduct backtests. Specifically, we employ a top-k/drop-n portfolio construction strategy, implemented on the Qlib platform (Yang et al. 2020). The practical trading performance is evaluated using two key metrics: Annualized Excess Return (AER), which measures the strategy's profitability, and Information Ratio (IR), which quantifies its risk-adjusted performance.

The combination of these four metrics (IC, RankIC, AER, and IR) provides a comprehensive evaluation of the mined alphas. As illustrated in Figure 4, our framework consistently outperforms baselines across all metrics. This demonstrates that the alphas mined by our framework possess superior predictive capabilities for future stock returns, which can be effectively translated into trading profitability.

Experiment 2: Ablation Study

We conduct ablation studies to evaluate three key components of our framework: MCTS, multi-dimensional feedback, and FSA.

Table 1 presents the impact of these components on predictive performance. When incorporating Effectiveness and Diversity as feedback, MCTS demonstrates superior predictive performance over CoT and ToT. Performance progressively improves with the integration of additional feedback dimensions. Notably, while Turnover feedback slightly reduces IC and RankIC, it enhances practical trading metrics (AER, IR) by mitigating transaction costs. The integration of FSA yields further improvements across all metrics for both LightGBM and MLP models. These results underscore the individual and collective contributions of MCTS, multi-dimensional feedback, and FSA to our framework's efficacy.

Beyond predictive performance, we analyze the search dynamics in Figure 5. Subplots (a) and (b) assess search efficiency by plotting the number of effective alphas mined against the total generated alphas. Our framework, even without FSA, demonstrates higher search efficiency than other LLM-based methods. FSA further amplifies this advantage. Furthermore, subplots (c) and (d) evaluate the quality and generalization of the top 50 alphas (selected by in-sample RankIC) over generations. The in-sample performance of our method's alphas improves as the search progresses (c). More importantly, this improvement translates to superior out-of-sample performance (d), indicating that our framework discovers more generalizable alphas.

To establish a more standardized and equitable basis for comparison, we present a cost-performance analysis in Figure 6. We estimate the cost of a single experimental run for each method by unifying its runtime and API usage into a monetary value based on public cloud computing prices. The analysis reveals that our framework achieves a favorable risk-return profile, with its variants occupying the highest Information Ratio (IR) contours. Notably, the overall cost and performance of our framework are primarily driven by the choice of the underlying LLM. For instance, employing

Search Strategy	Included Evaluation Dimensions					LightGBM				MLP			
	Eff.	Div.	Turn.	Stab.	O.R.	IC	RankIC	AER	IR	IC	RankIC	AER	IR
CoT	✓	✓	×	×	×	0.0434	0.0395	0.0707	0.7461	0.0421	0.0393	0.0922	0.9962
ToT	✓	✓	×	×	×	0.0459	0.0427	0.0868	0.9337	0.0452	0.0435	0.0945	1.0348
MCTS	✓	×	×	×	×	0.0409	0.0374	0.0941	0.9775	0.0400	0.0376	0.0935	1.0010
MCTS	✓	✓	×	×	×	0.0501	0.0476	0.1003	1.0106	0.0486	0.0462	0.1023	1.0462
MCTS	✓	✓	✓	×	×	0.0492	0.0457	0.1063	1.1062	0.0489	0.0462	0.1185	1.2556
MCTS	✓	✓	✓	✓	×	0.0495	0.0462	0.1030	1.0331	0.0491	0.0465	0.1093	1.1773
MCTS	✓	✓	✓	✓	✓	0.0515	0.0479	0.1075	1.1121	0.0503	0.0478	0.1166	1.2127
MCTS+FSA	✓	✓	✓	✓	✓	**0.0549**	**0.0512**	**0.1107**	**1.1792**	**0.0522**	**0.0503**	**0.1234**	**1.2712**

Table 1: Ablation study of our framework's components. Best results are highlighted in **bold**. We ablate across five evaluation dimensions: Effectiveness (Eff.), Diversity (Div.), Turnover (Turn.), Stability (Stab.), and Overfitting Risk (O.R.). The checkmark (✓) indicates a dimension is included in the search, while the cross (×) indicates it is not.

a lightweight model like Gemini-2.0-flash-lite yields a high IR of 1.27 at a minimal cost of $7.5. In contrast, using the more powerful GPT-4.1 results in a slightly lower IR of 1.23 at a substantially higher cost of $74.4. This highlights that our framework offers the flexibility to select an appropriate LLM, enabling a desirable balance between performance and computational budget.

Experiment 3: Interpretablitity of Alpha Formulas

In this experiment, we evaluate the interpretability of alpha formulas mined by different methods. We define the interpretability of an alpha formula by its capacity to articulate a reasonable logic, a specific market phenomenon, or an investment strategy. To quantify this, we randomly select one alpha formula per method and employ LLMs to rank their interpretability. We repeat this process 50 times and compute the average rank for each method. To mitigate potential biases from a single LLM, we aggregate rankings from three distinct LLMs. The results are presented in Figure 7. The findings indicate that formulas mined by our framework exhibit interpretability second only to those generated by the CoT method. Notably, they are consistently ranked as more interpretable than those from non-LLM baselines. This suggests that our approach achieves a compelling trade-off, delivering strong predictive performance while maintaining a high degree of interpretability.

While a formal human study is outside the scope of this work, a qualitative inspection of these examples reveals a clear difference: formulas from our method tend to have more discernible logic compared to the often opaque structures produced by non-LLM methods.

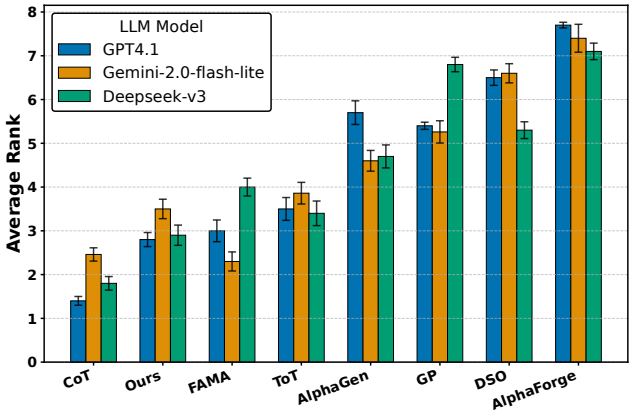

Figure 7: Interpretability Ranking Comparison, showing mean and standard deviation over 5 random seeds.

Related Work

Automated formulaic alpha mining has traditionally relied on genetic programming (GP) frameworks (Lin et al. 2019; Zhang et al. 2020), alongside reinforcement learning (Yu et al. 2023) and deep learning-based generative models (Shi et al. 2024). More recently, LLMs have been introduced, with approaches like FAMA (Li et al. 2024b) and AlphaAgent (Tang et al. 2025) leveraging them for direct alpha generation guided by in-context examples or heuristics. In contrast, our method frames alpha discovery as a formal reasoning task, uniquely employing MCTS to systematically explore the structured space of mathematical formulas. This approach is inspired by recent advancements in tree search-based reasoning, such as Tree of Thoughts (ToT) (Yao et al. 2024) and RethinkMCTS (Li et al. 2024a), which enhance LLM planning by structuring generation as a search process. A further literature review is provided in Appendix A.

Conclusion

We introduce an LLM-Powered Monte Carlo Tree Search (MCTS) framework for formulaic alpha mining. This approach models alpha mining as a tree search, where an LLM iteratively generates and refines candidate formulas, critically guided by quantitative feedback from financial backtesting. To foster search efficiency and alpha effectiveness, we incorporate a Frequent Subtree Avoidance mechanism. Experimental results demonstrate that our framework mines alphas with superior predictive accuracy and trading performance, while also offering enhanced interpretability and search efficiency compared to existing methods. This work pioneers a promising direction for leveraging LLMs and MCTS to tackle the complex challenge of automated formulaic alpha mining in finance.

Acknowledgements

This work was supported by the Xiongan AI Institute. We extend our gratitude to Zongliang Fu for insightful discussions and meticulous proofreading.

References

Coulom, R. 2007. Monte-Carlo Tree Search in Crazy Stone,". In *Proc. Game Prog. Workshop, Tokyo, Japan*, 74–75.

Dainese, N.; Merler, M.; Alakuijala, M.; and Marttinen, P. 2024. Generating code world models with large language models guided by monte carlo tree search. *arXiv preprint arXiv:2405.15383*.

DeLorenzo, M.; Chowdhury, A. B.; Gohil, V.; Thakur, S.; Karri, R.; Garg, S.; and Rajendran, J. 2024. Make every move count: Llm-based high-quality rtl code generation using mcts. *arXiv preprint arXiv:2402.03289*.

Duan, Y.; Wang, L.; Zhang, Q.; and Li, J. 2022. Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 4468–4476.

Fama, E. F.; and French, K. R. 1992. The cross-section of expected stock returns. *the Journal of Finance*, 47(2): 427–465.

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

Hou, K.; Xue, C.; and Zhang, L. 2020. Replicating anomalies. *The Review of financial studies*, 33(5): 2019–2133.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, 282–293. Springer.

Landajuela, M.; Lee, C. S.; Yang, J.; Glatt, R.; Santiago, C. P.; Aravena, I.; Mundhenk, T.; Mulcahy, G.; and Petersen, B. K. 2022. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems*, 35: 33985–33998.

Li, Q.; Xia, W.; Du, K.; Dai, X.; Tang, R.; Wang, Y.; Yu, Y.; and Zhang, W. 2024a. RethinkMCTS: Refining Erroneous Thoughts in Monte Carlo Tree Search for Code Generation. *arXiv preprint arXiv:2409.09584*.

Li, S.; Dong, S.; Luan, K.; Di, X.; and Ding, C. 2025. Enhancing Reasoning through Process Supervision with Monte Carlo Tree Search. *arXiv preprint arXiv:2501.01478*.

Li, Z.; Song, R.; Sun, C.; Xu, W.; Yu, Z.; and Wen, J.-R. 2024b. 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*, 3891–3902.

Lin, X.; Chen, Y.; Li, Z.; and He, K. 2019. Stock Alpha Mining Based On Genetic Algorithm. Technical report, Huatai Securities Research Center.

Qian, E. E.; Hua, R. H.; and Sorensen, E. H. 2007. *Quantitative equity portfolio management: modern techniques and applications*. Chapman and Hall/CRC.

Shi, H.; Song, W.; Zhang, X.; Shi, J.; Luo, C.; Ao, X.; Arian, H.; and Seco, L. 2024. AlphaForge: A Framework to Mine and Dynamically Combine Formulaic Alpha Factors. *arXiv preprint arXiv:2406.18394*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Tang, Z.; Chen, Z.; Yang, J.; Mai, J.; Zheng, Y.; Wang, K.; Chen, J.; and Lin, L. 2025. AlphaAgent: LLM-Driven Alpha Mining with Regularized Exploration to Counteract Alpha Decay. *arXiv preprint arXiv:2502.16789*.

Tulchinsky, I. 2019. *Finding Alphas: A quantitative approach to building trading strategies*. John Wiley & Sons.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xie, Y.; Goyal, A.; Zheng, W.; Kan, M.-Y.; Lillicrap, T. P.; Kawaguchi, K.; and Shieh, M. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*.

Xu, W.; Liu, W.; Wang, L.; Xia, Y.; Bian, J.; Yin, J.; and Liu, T.-Y. 2021a. Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. *arXiv preprint arXiv:2110.13716*.

Xu, W.; Liu, W.; Xu, C.; Bian, J.; Yin, J.; and Liu, T.-Y. 2021b. Rest: Relational event-driven stock trend forecasting. In *Proceedings of the web conference 2021*, 1–10.

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

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Yu, S.; Xue, H.; Ao, X.; Pan, F.; He, J.; Tu, D.; and He, Q. 2023. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5476–5486.

Yu, Y.; Li, H.; Chen, Z.; Jiang, Y.; Li, Y.; Zhang, D.; Liu, R.; Suchow, J. W.; and Khashanah, K. 2024. FinMem: A performance-enhanced LLM trading agent with layered memory and character design. In *Proceedings of the AAAI Symposium Series*, volume 3, 595–597.

Zhang, D.; Huang, X.; Zhou, D.; Li, Y.; and Ouyang, W. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*.

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

Zhang, T.; Zhang, Z. A.; Fan, Z.; Luo, H.; Liu, F.; Liu, Q.; Cao, W.; and Jian, L. 2023. OpenFE: automated feature generation with expert-level performance. In *International Conference on Machine Learning*, 41880–41901. PMLR.

Zhao, Z.; Lee, W. S.; and Hsu, D. 2023. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36: 31967–31987.