COSMOSALPHA: MULTI-RELATIONAL GRAPH SEARCH WITH PARETO-GUIDED EXPLORATION FOR LLM-DRIVEN ALPHA MINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering predictive alpha factors from noisy financial data is a fundamental yet challenging problem in quantitative finance. LLM-driven formulaic alpha mining offers an attractive paradigm by combining interpretable symbolic expressions with iterative generation and refinement guided by quantitative feedback. However, existing generation-centric and tree-based search strategies primarily organize factors through generation lineage, leaving structural and statistical relationships across search branches underutilized. Consequently, information remains localized along parent–child trajectories, which can confine exploration to redundant regions. Moreover, effective factor discovery must account for multiple distinct objectives that cannot be adequately captured by a single search criterion. We introduce **CosmosAlpha**, a graph-based LLM factor-mining framework that represents discovered factors as a multi-relational graph encoding generation lineage, expression-structure similarity, and cross-sectional exposure correlation. A Graph Flow mechanism propagates search utilities across the graph, allowing each factor to be assessed using relational evidence from nearby factors rather than in isolation. Pareto-guided selection jointly considers quality, fertility and diversity, explicitly modeling their trade-offs within a multi-objective search framework. Across CSI 300 and CSI 500, CosmosAlpha-Graph achieves the best mean rank over seven predictive and portfolio metrics and outperforms its controlled CosmosAlpha-MCTS counterpart on every reported metric, showing a consistent advantage for relational graph search in LLM-driven factor mining.

1 INTRODUCTION

Alpha mining seeks to discover quantitative signals from market data for predicting future stock returns (Jensen, 1968; Fama & MacBeth, 1973; Gu et al., 2020). In modern quantitative research, an *alpha factor* is a predictive signal used to rank stocks by their expected future returns; its predictive value is commonly assessed by the alignment between factor scores and subsequent cross-sectional returns and by the performance of portfolios formed from those scores (Fama & French, 1992; Gu et al., 2020). In formulaic alpha mining, the factor is represented as an explicit mathematical expression over observable market variables, making its construction and evaluation transparent (Kakushadze, 2016; Yu et al., 2023). Such expressions can capture diverse patterns in price, volume, and other market data. Established handcrafted libraries such as Alpha158 provide fixed collections of interpretable factors and serve as reference points for automated mining (Yang et al., 2020). Individual factors, whether handcrafted or automatically mined, are typically weak and noisy. Reliable discovery therefore requires searching a large expression space while controlling spurious correlations and redundant candidates (Harvey et al., 2016; Gu et al., 2020).

Automated alpha mining has progressively expanded both the symbolic search space and the feedback used to navigate it. Evolutionary approaches, including genetic programming (GP), search formula structures through mutation and selection; AutoAlpha and AlphaEvolve are representative examples of this direction (Zhang et al., 2020; Cui et al., 2021), while reinforcement-learning methods optimize expression construction or the quality of an emerging factor set (Yu et al., 2023; Zhao et al., 2025a;b; Zhu & Zhu, 2025). Large language models provide another route by introducing financial hypotheses, semantic reasoning, and iterative refinement from quantitative feedback (Li

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

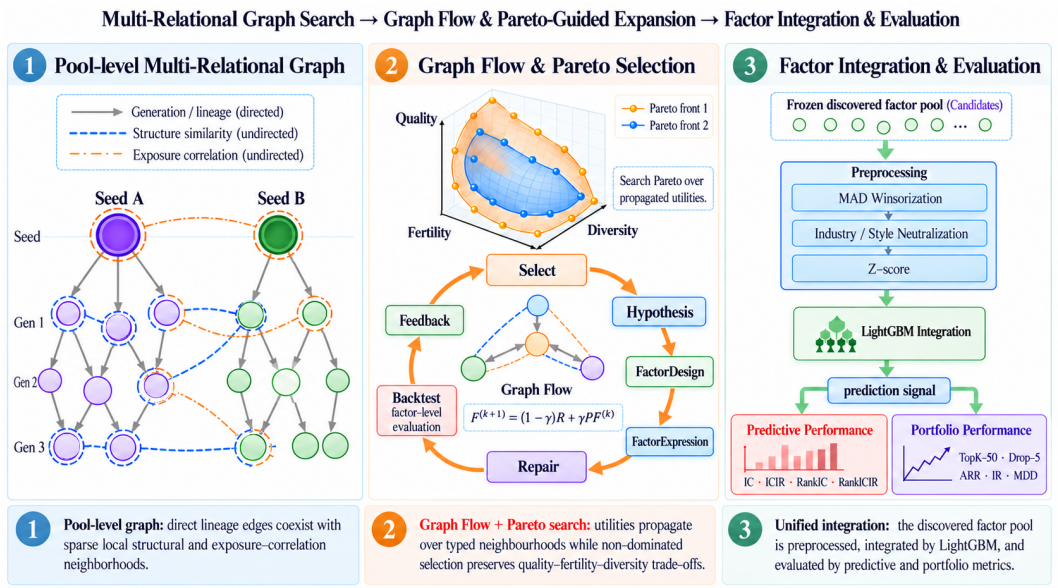


Figure 1: Overview of CosmosAlpha. (1) Evaluated factors form a pool-level multi-relational graph with generation, structure, and exposure-correlation edges. (2) Graph Flow propagates Quality, Fertility, and Diversity, after which Pareto-guided selection chooses parents for LLM-driven mutation and evaluated children return to the graph. (3) The fixed factor pool is preprocessed and integrated with LightGBM for common predictive and portfolio evaluation.

et al., 2024; Yu et al., 2026; Tang et al., 2025; Cao et al., 2025; Luo et al., 2026). Among recent LLM-based approaches, several systems also retain richer persistent search states: Alpha Jungle organizes refinements with Monte Carlo Tree Search (MCTS) (Shi et al., 2026), AlphaPROBE navigates a lineage-based directed acyclic graph (DAG) (Guo et al., 2026), QuantaAlpha evolves complete mining trajectories (Han et al., 2026), and AlphaAgentEvo learns multi-turn evolution through agentic reinforcement learning (Tang et al., 2026). Together, these developments move formulaic alpha mining from isolated generation toward iterative search that can reuse previous evaluations.

Despite this progress, many recent LLM-based methods still organize persistent search state primarily around provenance or trajectory. A generation edge records *who generated whom*, but it does not reveal whether factors from otherwise distant search paths reuse similar subexpressions or induce nearly identical exposure profiles across stocks. LLM-based systems address parts of this problem in different ways: AlphaAgent regularizes originality at the abstract syntax tree (AST) level, Alpha Jungle suppresses frequent subtrees, and AlphaPROBE reasons over global lineage (Tang et al., 2025; Shi et al., 2026; Guo et al., 2026). In a complementary GFlowNet-based approach, AlphaSAGE uses a structure-aware encoder for each expression's AST while promoting diverse generation (Chen et al., 2026). These approaches expose complementary signals, but they do not jointly treat generation history, structure similarity, and empirical cross-sectional exposure correlation as coexisting relation types between *complete evaluated factors*. As a result, evidence obtained along one search path remains difficult to reuse for otherwise distant factors that share similar structures or cross-sectional exposure profiles, which can keep exploration localized in redundant regions.

Selecting which factors to expand remains a difficult part of automated alpha mining. We argue that a candidate parent should be assessed along three complementary dimensions: current quality, fertility, and diversity. Current *quality* measures a factor's present predictive value; *fertility* measures whether its previous expansions produced high-quality descendants; and *diversity* measures whether it occupies a less redundant region of the factor pool. Existing methods often either prioritize quality alone or reduce these criteria to a fixed scalar, which can obscure meaningful trade-offs among candidate parents. Structure similarity and cross-sectional exposure correlation provide direct measures of redundancy for diversity estimation, while generation links preserve search provenance for rela-

tional propagation. Pareto-based multi-objective selection can then retain non-dominated trade-offs without collapsing these criteria to a single scalar (Zitzler & Thiele, 1999; Deb et al., 2002).

We introduce **CosmosAlpha**, an LLM-driven framework that treats the evaluated factor pool as an active multi-relational search state (Figure 1). Each node is a complete factor, and three relation types capture generation lineage, expression structure, and cross-sectional exposure correlation among factors. Graph Flow propagates node utilities through these typed neighborhoods so that parent assessment reflects both local measurements and evidence from related factors. Pareto-guided selection then preserves trade-offs among quality, fertility, and diversity, and the selected factors serve as parents for subsequent LLM-guided mutations. After evaluation, valid children enter the graph and update the state used by the next round. The factor archive is thus no longer a passive record of past attempts; it becomes the state that organizes subsequent exploration.

We evaluate CosmosAlpha on CSI 300 and CSI 500 under a common data and downstream LightGBM integration protocol. CosmosAlpha-Graph achieves the best overall mean rank on both universes and outperforms the controlled CosmosAlpha-MCTS counterpart on every reported predictive and portfolio metric. Component and relation ablations support the contributions of relational propagation and multi-objective selection, while experiments across three LLM backbones and three independent runs show that the overall advantage is not confined to a single model or stochastic run.

Our contributions are summarized as follows:

- We formulate LLM-driven formulaic alpha mining as iterative expansion over a *pool-level multi-relational graph* that jointly represents generation lineage, expression structure, and empirical exposure correlation between complete factors.
- We develop Graph Flow and Pareto-guided expansion to combine relational evidence with distinct quality, fertility, and diversity criteria, preserving useful search trade-offs without fixed scalarization.
- We provide a controlled evaluation in which CosmosAlpha-Graph and CosmosAlpha-MCTS share the factor language, evaluator, LLM backbone, and downstream modeling protocol, thereby isolating the effect of relational graph search; cross-universe comparisons, ablations, LLM sensitivity, and multi-run analysis further test the robustness of the design.

2 Related Work

Formulaic alpha mining. Automated formulaic alpha mining has progressed from evolutionary search to methods that optimize factor sets and use learned feedback. Genetic programming systems such as AutoAlpha use hierarchical evolution and quality-diversity guidance (Zhang et al., 2020), while AlphaEvolve expands formula representations and prunes redundant candidates (Cui et al., 2021). Reinforcement-learning and feedback-driven generative methods instead explore factor-set construction and richer feedback mechanisms. AlphaGen and AlphaForge optimize factor-set construction, while QuantFactor REINFORCE and trajectory-level reward shaping improve the stability and density of RL signals; RiskMiner and AlphaQCM explore alternative tree-search and distributional-RL objectives (Yu et al., 2023; Shi et al., 2025; Zhao et al., 2025a;b; Ren et al., 2024; Zhu & Zhu, 2025). These approaches broaden formula generation and factor-set construction, but generally do not represent the complete evaluated factor pool as a state with explicit relations among its members.

LLM-driven and agentic alpha search. LLM-based systems add semantic hypothesis generation and feedback-driven refinement to formula mining. FAMA and recent hybrid LLM–RL frameworks integrate LLM proposals with symbolic or optimization loops, while AlphaAgent and Chain-of-Alpha organize generation and feedback through multi-agent or structured generation-and-feedback chains (Li et al., 2024; Yu et al., 2026; Tang et al., 2025; Cao et al., 2025). AlphaBench provides a systematic evaluation suite across factor generation, evaluation, and search (Luo et al., 2026). More structured systems preserve search history explicitly: Alpha Jungle uses MCTS, QuantaAlpha evolves complete trajectories, and AlphaAgentEvo learns multi-turn evolution with agentic reinforcement learning (Kocsis & Szepesvári, 2006; Shi et al., 2026; Han et al., 2026; Tang et al., 2026). R&D-Agent-Quant extends the scope to coordinated factor and model R&D (Li et al., 2025).

These methods retain richer workflow state, but it remains tied to a particular trajectory or task-specific agent state rather than a shared factor-level representation. The remaining question is how the pool itself can serve as a persistent, relational state for formula search.

Relational graphs and multi-objective search. The closest structural precedents represent either factor lineage or an individual expression. AlphaPROBE represents complete factors in a lineage-based DAG for retrieval and biased evolution (Guo et al., 2026); AlphaSAGE encodes the AST of an individual expression with a structure-aware graph encoder and uses GFlowNets to promote diverse high-reward generation (Bengio et al., 2021; Chen et al., 2026). Graph Flow is conceptually related to diffusion-style propagation over graph neighborhoods, which shares information among locally connected nodes (Zhou et al., 2003; Klicpera et al., 2019). CosmosAlpha applies this principle over complete evaluated factors, with distinct relations for generation provenance, structure similarity, and cross-sectional exposure correlation, rather than learning an expression encoder or a generative flow policy. This factor-pool view jointly exposes relation types that prior methods handle separately, enabling the same state to support both information propagation and parent selection.

Multi-objective evolutionary optimization provides the complementary selection principle. Pareto archives and diversity-preserving selection maintain solutions with different trade-offs instead of imposing a single weighted objective (Zitzler & Thiele, 1999; Deb et al., 2002). CosmosAlpha applies this principle to propagated quality, fertility, and diversity, separating graph-based evidence sharing from parent selection and preserving non-dominated trade-offs during parent selection.

3 METHOD

3.1 PROBLEM FORMULATION AND OVERVIEW

We consider formulaic alpha discovery over a time-varying stock universe. Let $\mathcal{X}_t \in \mathbb{R}^{N_t \times d}$ denote the market observations available on day t, and let a formulaic factor f_i produce a cross-sectional exposure vector $\mathbf{z}_{i,t} = f_i(\mathcal{X}_{\leq t})$. Given a finite search budget, the objective is to discover a factor library that is predictive while avoiding repeated exploration of redundant regions.

Existing LLM-based alpha-mining methods commonly organize search as sequential refinement, tree search, or generation-centric evolution (Tang et al., 2025; Shi et al., 2026; Guo et al., 2026). These representations primarily preserve *who generated whom*, but factors from different search branches may still share similar expressions or induce highly correlated stock exposures. CosmosAlpha therefore models factor discovery as search over a multi-relational graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\text{gen}}, \mathcal{E}^{\text{str}}, \mathcal{E}^{\text{corr}}), \tag{1}$$

where nodes are evaluated factors and the three edge types encode generation lineage, expression structure, and empirical exposure correlation. Each node is scored along quality, fertility, and diversity; these utilities are propagated through the graph by Graph Flow, and Pareto-guided selection chooses parents for LLM mutation. Valid children are evaluated and inserted back into the graph, yielding an iterative graph-search process.

3.2 MULTI-RELATIONAL FACTOR GRAPH

The three relations capture complementary notions of factor proximity.

Generation relation. If factor f_j is produced by mutating factor f_i, we record a directed provenance edge $i \to j$. For message propagation we use the symmetrized direct parent–child adjacency

$$W^{\text{gen}}_{ij} = \mathbb{I}[i \text{ and } j \text{ are a direct parent–child pair}]. \tag{2}$$

Longer-range lineage information is captured by multi-step propagation rather than explicit ancestor–descendant edges.

Structure relation. Let $\mathcal{S}_i$ be the set of canonical subexpressions extracted from the expression tree of f_i. We define

$$S^{\text{str}}_{ij} = \frac{|\mathcal{S}_i \cap \mathcal{S}_j|}{|\mathcal{S}_i \cup \mathcal{S}_j|}, \qquad d^{\text{str}}_{ij} = 1 - S^{\text{str}}_{ij}. \tag{3}$$

Canonicalization preserves operator, topology, field, and parameter identities while removing spurious differences caused by commutative argument ordering.

Correlation relation. Expression structure does not fully determine empirical factor behavior. We therefore measure behavioral proximity by first computing the daily cross-sectional Pearson correlation and then averaging equally over valid dates:

$$C_{ij} = \frac{1}{|\mathcal{T}_{ij}|} \sum_{t \in \mathcal{T}_{ij}} \mathrm{Pearson}_{\mathrm{cs}}(\mathbf{z}_{i,t}, \mathbf{z}_{j,t}), \qquad d_{ij}^{\mathrm{corr}} = 1 - |C_{ij}|. \tag{4}$$

The absolute value treats strongly negative and strongly positive exposure correlations as equally redundant up to factor orientation. The structure and correlation graphs retain only local neighbors; exact construction rules are reported in Appendix A.3.

Let W^m denote the weighted adjacency matrix for relation $m \in \{\mathrm{gen}, \mathrm{str}, \mathrm{corr}\}$. For the retained structure and correlation neighbors, the edge weights are given by the corresponding similarities: $W_{ij}^{\mathrm{str}} = S_{ij}^{\mathrm{str}}$ and $W_{ij}^{\mathrm{corr}} = |C_{ij}|$. Each modality is row-normalized as $P^m = (D^m)^{-1} W^m$, where $D_{ii}^m = \sum_j W_{ij}^m$. We combine the three modalities as

$$\boxed{P = \frac{1}{3}\left(P^{\mathrm{gen}} + P^{\mathrm{str}} + P^{\mathrm{corr}}\right).} \tag{5}$$

Separate normalization prevents denser similarity graphs from overwhelming the sparser generation graph.

3.3 Multi-Objective Search Utility

A useful parent should not be judged only by its current predictive strength: it may also be valuable because it repeatedly generates strong descendants or because it occupies an underexplored region. CosmosAlpha therefore assigns three complementary utilities to each node.

Intrinsic quality. We characterize standalone factor quality by

$$\mathbf{q}_i = \left(|\overline{\mathrm{RankIC}}_i|, -\sigma(\mathrm{RankIC}_i), R_i^{\mathrm{Top20}}\right), \tag{6}$$

combining predictive strength, temporal stability, and a direction-aligned Top-20% return metric. A *Quality Pareto Archive* retains the non-dominated factors under these raw objectives. For scalar node quality, the three objectives are mapped through fixed pre-specified normalization functions and averaged to obtain Q_i^{self}. The method-specific normalization and archive semantics are given in Appendix A.4.

Fertility. Current quality does not measure whether a node reliably produces useful descendants. Let $\mathcal{A}_r^-$ denote the Quality Pareto Archive frozen before round r. For a newly evaluated child c, its birth-time quality contribution is

$$\boxed{\Delta HV_c = HV(\mathcal{A}_r^- \cup \{c\}) - HV(\mathcal{A}_r^-).} \tag{7}$$

All children in the same round use the same frozen archive, avoiding order-dependent rewards. The fertility of parent i, denoted Q_i^{fert}, aggregates its best historical expansion outcomes, so it measures the node's demonstrated ability to produce frontier-expanding descendants rather than its current quality alone. The exact attempt-level aggregation is specified in Appendix A.4.

Exploration diversity. At the beginning of round r, we select a high-quality reference set $\mathcal{R}_r$ from the current Quality Pareto Archive and keep it fixed while computing diversity for that round. To discourage concentration around a few successful factor families, we measure novelty relative to this set. Using the structure and correlation distances defined above,

$$Q_i^{\mathrm{div}} = \frac{1}{2}\left[\mathrm{AvgKNN}_{j \in \mathcal{R}_r} d_{ij}^{\mathrm{str}} + \mathrm{AvgKNN}_{j \in \mathcal{R}_r} d_{ij}^{\mathrm{corr}}\right]. \tag{8}$$

Generation distance is excluded to avoid double-counting lineage information already represented in the graph topology. Reference-set construction is reported in Appendix A.4.

The resulting node utility is the three-dimensional vector

$$\boxed{\mathbf{R}_i = (Q_i^{\text{self}}, Q_i^{\text{fert}}, Q_i^{\text{div}}).} \tag{9}$$

We deliberately preserve these objectives instead of assigning a fixed scalar trade-off among current quality, fertility, and diversity.

3.4 Graph Flow and Pareto-Guided Expansion

The raw utility $\mathbf{R}_i$ reflects node-specific evidence but ignores the quality of its neighborhood. The transition matrix P in Eq. 5 represents the generation, structure, and correlation links between factors and sets the weights for aggregating neighboring utilities. Stacking node utilities into R, Graph Flow iterates

$$\boxed{F^{(k+1)} = (1-\gamma)R + \gamma P F^{(k)}, \qquad F^{(0)} = R,} \tag{10}$$

where $\gamma \in [0, 1)$ controls the contribution of neighboring nodes. The first term preserves node-specific evidence, while the second propagates information through generation, structure, and correlation neighborhoods. Setting $\gamma = 0$ removes graph propagation and yields a natural ablation.

Parent selection operates directly on the propagated vectors $\mathbf{F}_i = (F_i^{\text{self}}, F_i^{\text{fert}}, F_i^{\text{div}})$. We apply non-dominated sorting rather than a manually weighted scalarization; parents are drawn from successive Pareto fronts, and crowding distance resolves a front that exceeds the remaining parent budget. This *search Pareto* mechanism is distinct from the intrinsic Quality Pareto Archive: the former controls exploration in propagated utility space, whereas the latter measures factor-quality frontier improvement for ΔHV.

For each selected parent, the LLM receives its factor expression and search context and generates parent-grounded mutations. Valid unique children are evaluated, inserted into the graph, and used to update the Quality Pareto Archive, parent fertility, and all affected graph relations. The complete round semantics, budget rules, and pseudocode are provided in Appendix A.7.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate factor mining on the CSI 300 and CSI 500 universes using daily A-share market data. Historical constituent membership and trading-eligibility masks are applied point-in-time, and all methods use the same data snapshot and preprocessing conventions. We use three chronological, non-overlapping segments. The training set (April 1, 2010–May 31, 2019) provides feedback for factor mining and is subsequently used to fit the downstream LightGBM model. The validation set (June 1, 2019–December 31, 2020) is used only for LightGBM early stopping. The test set (January 1, 2021–December 31, 2025) is held out until the factor pools and fitted model configurations are fixed, and is used only to compute final out-of-sample predictions and portfolio results. Neither the validation nor test set is used to select factor expressions or tune the search methods.

Baselines and mining protocol. We compare against the handcrafted Alpha158 factor library (Yang et al., 2020), genetic programming (GP) (Koza, 1992), and representative recent automated factor-mining methods. The LLM-driven baselines include AlphaAgent (Tang et al., 2025), R&D-Agent (Li et al., 2025), AlphaPROBE (Guo et al., 2026), and QuantaAlpha (Han et al., 2026); we additionally evaluate AlphaGen (Yu et al., 2023), AlphaForge (Shi et al., 2025), and AlphaSAGE (Chen et al., 2026) as recent factor-collection and structure-aware search methods. We further include CosmosAlpha-MCTS as a controlled lineage-based search baseline using the same factor language and evaluation stack as our graph variant. AlphaJungle (Shi et al., 2026) is not included as a separate row because we did not identify an official public implementation at the time of submission; CosmosAlpha-MCTS therefore serves as a transparent MCTS-based proxy. It

Table 1: Performance on CSI 300. All metrics are computed on the held-out test set after LightGBM integration of each method's factor pool. ARR and signed MDD are percentages; the remaining metrics are on their native scales. The final column averages ranks over the seven metrics (rank 1 is best; ties receive average ranks). Best and second-best results in each column are bolded and underlined, respectively.

Method	Predictive Performance				Portfolio Performance			Overall
	IC ↑	ICIR ↑	RankIC ↑	RankICIR ↑	ARR (%) ↑	IR ↑	MDD (%) ↑	Mean rank ↓
Alpha158	<u>0.0320</u>	<u>0.3474</u>	<u>0.0479</u>	<u>0.5668</u>	6.32	0.3945	-29.73	<u>3.86</u>
GP	0.0231	0.3098	0.0300	0.4483	4.32	0.2215	-42.45	9.71
AlphaGen	0.0301	0.3128	0.0452	0.5461	4.25	0.2490	-32.47	7.29
AlphaForge	0.0304	0.3404	0.0469	0.5615	6.88	0.4164	-25.41	<u>3.86</u>
AlphaSAGE	0.0243	0.2574	0.0432	0.4750	6.57	0.4007	-27.50	6.57
AlphaAgent	0.0235	0.2724	0.0390	0.4753	7.13	<u>0.4393</u>	-26.54	6.00
R&D-Agent	0.0173	0.1811	0.0356	0.3945	5.99	0.3667	**-24.87**	8.43
QuantaAlpha	0.0211	0.2154	0.0393	0.4167	<u>7.22</u>	0.4303	-26.11	6.71
AlphaPROBE	0.0319	**0.3536**	0.0478	0.5652	6.20	0.3464	-35.12	5.00
CosmosAlpha-MCTS	0.0281	0.3238	0.0409	0.5145	5.68	0.3296	-31.56	7.14
CosmosAlpha-Graph (Ours)	**0.0324**	0.3441	**0.0494**	**0.5670**	**8.44**	**0.5332**	<u>-25.07</u>	**1.43**

maintains multiple subtrees rather than a single large tree, which encourages more diverse factor-generation trajectories while preserving the lineage-based search structure. Unless otherwise specified, DeepSeek-V4-Flash is used as the LLM backbone for LLM-driven methods. We align the mining budget so that each automated method produces a factor pool of roughly 150 evaluable factors for downstream integration. Each method retains its own search procedure; the factors in its resulting pool are then recomputed on the common data snapshot and evaluated with the shared preprocessing, LightGBM, and portfolio protocol. Appendix C.7 describes the baselines and lists available public implementation repositories.

Evaluation protocol and metrics. To measure the information contained in a discovered factor *set*, we recompute each method's factor pool under the common data protocol and use the factors jointly as LightGBM features. The same preprocessing, model, and backtest configuration is used for every method; metric definitions and implementation details are given in Appendices B and C, respectively. Predictive performance is measured from held-out LightGBM predictions using IC, ICIR, RankIC, and RankICIR. Portfolio performance is measured from the same signal using annualized return (ARR), information ratio (IR), and signed maximum drawdown (MDD). Higher is better for the first six metrics; because MDD is reported as a negative return, a value closer to zero is better.

4.2 Main Results

Tables 1 and 2 compare the integrated factor pools on CSI 300 and CSI 500, respectively. On CSI 300, CosmosAlpha-Graph attains the best mean rank (1.43), leading IC, RankIC, RankICIR, ARR, and IR. AlphaPROBE leads ICIR, while R&D-Agent has the best signed MDD (-24.87%) and CosmosAlpha-Graph is second (-25.07%). Alpha158 and AlphaForge tie for the next-best mean rank (3.86). CosmosAlpha-Graph still exceeds the controlled CosmosAlpha-MCTS baseline on all seven metrics, which isolates the benefit of changing the search organization while retaining a shared factor language and evaluation stack.

On CSI 500, CosmosAlpha-Graph attains the best mean rank (1.29) and leads IC, ICIR, RankIC, ARR, and signed MDD, while AlphaAgent leads IR. Alpha158 remains strongest on RankICIR. CosmosAlpha-MCTS is the best non-Graph method by mean rank (4.14) and is also the strongest non-Graph method on IC (0.0356), followed by Alpha158 at 4.43. These results indicate that the graph variant's advantage also extends to CSI 500, although Alpha158 retains an advantage on RankICIR.

Table 2: Performance on CSI 500 under the protocol of Table 1. All rows report final results. The final column averages ranks over the seven metric columns (rank 1 is best; ties receive average ranks). Best and second-best results in each column are bolded and underlined, respectively; for signed MDD, a value closer to zero is better.

Method	Predictive Performance				Portfolio Performance			Overall
	IC ↑	ICIR ↑	RankIC ↑	RankICIR ↑	ARR (%) ↑	IR ↑	MDD (%) ↑	Mean rank ↓
Alpha158	0.0339	<u>0.3927</u>	<u>0.0618</u>	**0.7898**	11.07	0.5856	-28.41	4.43
GP	0.0213	0.2752	0.0438	0.6092	12.52	0.6101	-38.06	8.57
AlphaGen	0.0296	0.3613	0.0573	0.7147	11.82	0.6077	-34.26	6.43
AlphaForge	0.0284	0.3129	0.0590	0.7076	11.08	0.5758	-28.70	6.64
AlphaSAGE	0.0245	0.2652	0.0531	0.6394	10.51	0.5533	-27.69	8.71
AlphaAgent	0.0227	0.2516	0.0522	0.5917	<u>12.76</u>	**0.6798**	<u>-26.51</u>	6.71
R&D-Agent	0.0236	0.2679	0.0543	0.6855	11.88	0.6239	-33.44	7.43
QuantaAlpha	0.0266	0.3221	0.0533	0.6629	12.16	0.6370	-29.20	6.07
AlphaPROBE	0.0309	0.3688	0.0585	0.7551	11.37	0.5937	-29.98	5.57
CosmosAlpha-MCTS	<u>0.0356</u>	0.3898	0.0590	0.7472	12.16	0.6258	-32.18	<u>4.14</u>
CosmosAlpha-Graph (Ours)	**0.0360**	**0.3997**	**0.0627**	<u>0.7562</u>	**13.01**	<u>0.6689</u>	**-26.42**	**1.29**

4.3 Ablation Study

We use the CSI 300 ablations to isolate five parts of the search: Graph Flow propagation, intrinsic-quality archive construction, parent ranking after Flow, utility composition, and graph relations. The propagation control sets $\gamma = 0$. Quality scalarization replaces the intrinsic-quality Pareto archive with an equal-weight scalar score; this replacement also changes the high-quality reference set and the Fertility computation used by the search. Flow scalarization keeps Graph Flow but replaces Pareto fronts and crowding distance for parent ranking with an equal-weight score over normalized propagated (Q, F, D). The utility controls remove Fertility, remove Diversity, or retain Quality alone, and the relation controls remove Generation, Structure, or Correlation individually. Here Q, F, and D denote SelfQuality, Fertility, and AverageDistance, while G, S, and C denote the three relations. All variants use the same data split, LLM and prompt configuration, search budget, and downstream evaluation protocol.

Every ablation in Table 3 falls below the full model on all seven metrics. This consistent pattern supports the joint use of the quality archive, Flow-level parent selection, all three utility objectives, and all three graph relations. Appendix H gives the detailed changes to the quality reference set, Fertility, Flow normalization, and relation handling.

4.4 Search Behavior Analysis

We examine how the search state evolves as the number of generated factors increases. Figure 2 compares node-level Quality, Fertility, and AverageDistance as the cumulative number of factor-generation steps increases. For MCTS, these quantities are computed post hoc for diagnostic comparison. Each point records these quantities for the newly generated factor at that step. All three quantities are higher-is-better signals.

The three panels reveal complementary differences between the search dynamics. During the first 20 steps, MCTS has higher Quality than Graph, but its trajectory subsequently becomes volatile and trends downward, whereas Graph improves steadily. MCTS also shows substantially higher Fertility at the beginning, indicating considerable early gains from its search process, but this signal then drops rapidly close to zero. Thus, after roughly 20 steps, most subsequent MCTS iterations produce little incremental improvement. In contrast, Graph maintains a stable Fertility signal throughout the search, indicating continued productive improvement. This sustained Fertility is consistent with Graph's continued Quality gains in the later stages. Finally, Graph has higher AverageDistance over nearly the entire trajectory. This pattern supports the role of the generation, structure, and correlation relations in preserving factor diversity and reducing the risk of local search. Overall, Graph steadily improves factor quality while preserving diversity.

Table 3: Ablation study on CSI 300 after LightGBM integration. Quality scalarization changes the intrinsic-quality archive, reference-set construction, and Fertility computation; Flow scalarization changes parent ranking after Graph Flow. Remaining variants remove the named utility or relation. Higher is better for all seven metrics, including signed MDD; bold indicates the best result in each column.

Variant	Predictive Performance				Portfolio Performance		
	IC ↑	ICIR ↑	RankIC ↑	RankICIR ↑	ARR (%) ↑	IR ↑	MDD (%) ↑
Full Graph: (Q, F, D); relations (G, S, C)	**0.0324**	**0.3441**	**0.0494**	**0.5670**	**8.44**	**0.5332**	**-25.07**
w/o Flow: $\gamma = 0$	0.0278	0.2929	0.0436	0.4937	5.90	0.3541	-28.73
Quality scalarization: intrinsic quality	0.0284	0.3107	0.0452	0.5117	5.80	0.3539	-27.13
Flow scalarization: normalized (Q, F, D)	0.0289	0.2987	0.0474	0.5201	5.80	0.3486	-26.86
w/o Fertility: Pareto (Q, D)	0.0297	0.3181	0.0456	0.5286	6.71	0.4132	-26.35
w/o Diversity: Pareto (Q, F)	0.0286	0.3119	0.0448	0.5252	6.95	0.4273	-26.16
Quality only: Pareto (Q)	0.0296	0.3099	0.0475	0.5340	5.64	0.3421	-27.33
w/o Generation relation	0.0298	0.3189	0.0470	0.5499	6.59	0.3907	-28.55
w/o Structure relation	0.0276	0.3022	0.0453	0.5289	4.37	0.2659	-29.27
w/o Correlation relation	0.0295	0.3290	0.0447	0.5304	7.28	0.4401	-26.06

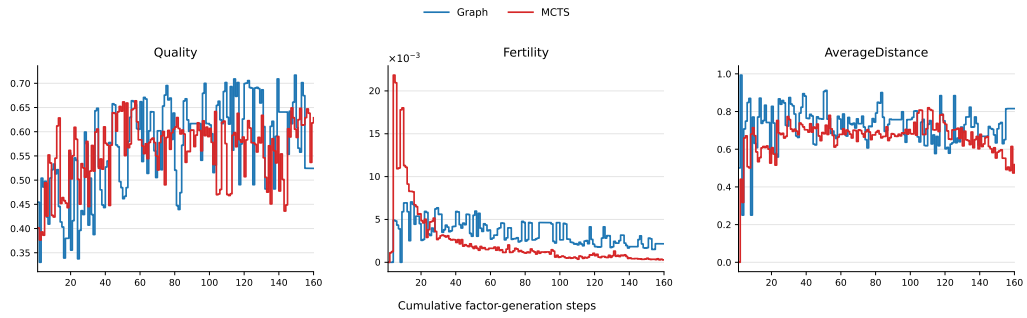


Figure 2: Search-state trajectories of CosmosAlpha-Graph and CosmosAlpha-MCTS over cumulative factor-generation steps. From left to right, the panels show node-level Quality, Fertility, and AverageDistance. Each point reports these quantities for the newly generated factor at the aligned diagnostic factor-generation index. Panels use native vertical scales.

5 Conclusion

We introduced CosmosAlpha, an LLM-driven framework that organizes evaluated factors as a multi-relational search state. It connects generation lineage, expression structure, and cross-sectional exposure correlation, propagates quality, fertility, and diversity through Graph Flow, and uses Pareto-guided selection to guide expansion beyond local generation history.

Under a common LightGBM integration protocol, CosmosAlpha-Graph achieves the best mean rank on both CSI 300 and CSI 500 and outperforms the controlled CosmosAlpha-MCTS variant on every reported metric. Every tested ablation underperforms the full model across all seven metrics, supporting relational propagation, Pareto-guided selection, utility composition, and the three graph relations. The LLM and independent-run studies further show that this advantage is not confined to one backbone or stochastic run. Future work will apply CosmosAlpha to additional markets and to the discovery of high-frequency and fundamental factors.

AI use statement

Generative AI tools were used primarily for English-language editing, grammar correction, and local improvements in clarity and organization. The research question, core ideas, method design,

experimental protocol, implementation, analysis, and conclusions were developed by the authors. The authors reviewed and verified all AI-assisted text, equations, citations, technical descriptions, and reported results, and take full responsibility for the final manuscript.

Ethics Statement

This work uses historical market data and does not involve human participants, personal data, or interaction with individual investors. The study is an offline research evaluation, and its backtest results should not be interpreted as financial advice or guaranteed live-trading performance. Automated factor discovery can encourage overfitting or unsafe deployment when used without independent validation and appropriate risk controls. We therefore report the temporal split and held-out test protocol and limit our claims to the evaluated stock universes and period.

Reproducibility Statement

Appendix C specifies the rolling-operator lookback window, temporal splits, factor-pool construction, preprocessing, LightGBM configuration, and portfolio protocol. It also states that factor mining and LightGBM early stopping use only the training and validation periods, respectively, while all reported metrics are computed on the held-out test period. Appendix I and Appendix J document the tested LLM identifiers and independent runs. The released artifacts will identify the exact code revision used for each experiment and provide the environment specification, prompts, model identifiers, run configurations, and scripts needed to regenerate the reported tables and figures.

References

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27381–27394, 2021.

Lang Cao, Zekun Xi, Long Liao, Ziwei Yang, and Zheng Cao. Chain-of-Alpha: Unleashing the power of large language models for alpha mining in quantitative trading. *arXiv preprint arXiv:2508.06312*, 2025. URL https://arxiv.org/abs/2508.06312.

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. 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.

Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. AlphaEvolve: A learning framework to discover novel alphas in quantitative investment. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2208–2216. Association for Computing Machinery, 2021. doi: 10.1145/3448016.3457324.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Eugene F. Fama and Kenneth R. French. The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465, 1992. doi: 10.1111/j.1540-6261.1992.tb04398.x.

Eugene F. Fama and James D. MacBeth. Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy*, 81(3):607–636, 1973. doi: 10.1086/260061.

Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020. doi: 10.1093/rfs/hhaa009.

Taian Guo, Haiyang Shen, Junyu Luo, Binqi Chen, Hongjun Ding, Jinsheng Huang, Luchen Liu, Yun Ma, and Ming Zhang. AlphaPROBE: Alpha mining via principled retrieval and on-graph biased evolution. *arXiv preprint arXiv:2602.11917*, 2026. URL https://arxiv.org/abs/2602.11917.

Jun Han, Shuo Zhang, Wei Li, Zhi Yang, Yifan Dong, Tu Hu, Jialuo Yuan, Xiaomin Yu, Yumo Zhu, Fangqi Lou, Xin Guo, Zhaowei Liu, Tianyi Jiang, Ruichuan An, Jingping Liu, Biao Wu, Rongze Chen, Kunyi Wang, Yifan Wang, Sen Hu, Xinbing Kong, Liwen Zhang, Ronghao Chen, and Huacan Wang. QuantaAlpha: An evolutionary framework for LLM-driven alpha mining. *arXiv preprint arXiv:2602.07085*, 2026. URL https://arxiv.org/abs/2602.07085.

Campbell R. Harvey, Yan Liu, and Heqing Zhu. ... and the Cross-Section of Expected Returns. *The Review of Financial Studies*, 29(1):5–68, 2016. doi: 10.1093/rfs/hhv059.

Michael C. Jensen. The performance of mutual funds in the period 1945–1964. *The Journal of Finance*, 23(2):389–416, 1968. doi: 10.1111/j.1540-6261.1968.tb00815.x.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016. doi: 10.1002/wilm.10525.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 2006. URL https://doi.org/10.1007/11871842_29.

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

Yuante Li, Xu Yang, Xiao Yang, Xisen Wang, Weiqing Liu, and Jiang Bian. R&D-Agent-Quant: A multi-agent framework for data-centric factors and model joint optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3967.

Zhiwei Li, Ran Song, Caihong Sun, Wei Xu, Zhengtao Yu, and Ji-Rong Wen. Can large language models mine interpretable financial factors more effectively? a neural-symbolic factor mining agent model. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3891–3902. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.233.

Haochen Luo, Ho Tin Ko, Jiandong Chen, David Sun, Yuan Zhang, and Chen Liu. 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.

Tao Ren, Ruihan Zhou, Jinyang Jiang, Jiafeng Liang, Qinghao Wang, and Yijie Peng. RiskMiner: Discovering formulaic alphas via risk seeking monte carlo tree search. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pp. 752–760. Association for Computing Machinery, 2024. doi: 10.1145/3677052.3698613.

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Angel Seco. AlphaForge: A framework to mine and dynamically combine formulaic alpha factors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12524–12532, 2025. doi: 10.1609/aaai.v39i12.33365.

Yu Shi, Yitong Duan, and Jian Li. Navigating the Alpha Jungle: An LLM-powered MCTS framework for formulaic alpha factor mining. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 997–1005, 2026. doi: 10.1609/aaai.v40i2.37069.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. AlphaAgent: LLM-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2813–2822. Association for Computing Machinery, 2025. doi: 10.1145/3711896.3736838.

Ziyi Tang, Xuexiong Yin, Weixin Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lNmZrawUMu.

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

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5476–5486. Association for Computing Machinery, 2023. doi: 10.1145/3580305.3599831.

Shuo Yu, Hong-Yan Xue, Xiang Ao, and Qing He. A hybrid approach to formulaic alpha discovery with large language model assistance. *Frontiers of Computer Science*, 20(2):2002316, 2026. doi: 10.1007/s11704-025-41061-5.

Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. AutoAlpha: An efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment. *arXiv preprint arXiv:2002.08245*, 2020. URL https://arxiv.org/abs/2002.08245.

Junjie Zhao, Chengxi Zhang, Min Qin, and Peng Yang. QuantFactor REINFORCE: Mining steady formulaic alpha factors with variance-bounded REINFORCE. *IEEE Transactions on Signal Processing*, 73:2448–2463, 2025a. doi: 10.1109/TSP.2025.3576781.

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

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pp. 321–328. MIT Press, 2003.

Zhoufan Zhu and Ke Zhu. 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, 2025. URL https://proceedings.mlr.press/v267/zhu25ag.html.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. doi: 10.1109/4235.797969.

A Additional Method Details

This appendix specifies the search semantics omitted from Section 3. We focus on choices that change graph state, node utility, or the search trajectory, and report mechanism-defining numerical settings where they clarify the algorithm. Metric definitions, prompt templates, and the factor-language catalog are kept separate from these algorithmic definitions. Appendix B defines the single-factor statistics used by the search.

A.1 Initialization and Node Admission

Each run begins with an initial set of factors proposed by the LLM. These proposals pass through the same expression validation, factor calculation, and evaluation pipeline as factors generated in later rounds. Once the initial factors have been admitted, CosmosAlpha constructs their relation matrices and initializes the Quality Pareto Archive before selecting any parent for expansion.

Graph admission is a technical-validity condition rather than a quality-selection rule. A candidate must have a valid expression, successfully computed factor values, a completed evaluation under the run's common data and preprocessing protocols, and finite values for mean RankIC, RankIC standard deviation, and direction-aligned Top-20% return. The evaluation must also contain at least two valid dates and cover at least 80% of the candidate evaluation dates. Candidates failing these conditions do not become graph nodes; quality comparisons are applied only after admission through the Quality Pareto Archive and parent-selection procedure.

A.2 Canonical Factor Identity and Duplicate Handling

CosmosAlpha assigns graph identity from a canonicalized expression under a fixed data frequency and factor language. Canonicalization recursively normalizes the abstract syntax tree (AST) and orders the operands of supported commutative operators listed in Table 15, while retaining operator identity, tree topology, fields, constants, and window arguments. It therefore removes equivalent operand orderings without attempting general algebraic simplification.

Duplicate checking occurs after expression validation and before factor calculation. Each validated candidate is compared with all committed graph nodes and with the other candidates proposed in the same batch. If several same-batch candidates have the same canonical identity, one is retained by a deterministic ordering and the rest are treated as duplicates. Expressions that have already been evaluated but failed the graph-admission conditions are likewise not reevaluated. A duplicate candidate creates neither a node nor a generation edge, but its parent expansion still consumes one attempt; an attempt with no valid unique child contributes zero to Fertility.

A.3 Multi-Relational Graph Construction

The three relation matrices are reconstructed over the currently admitted nodes. Generation records search provenance, whereas structure and correlation first define pairwise similarities and then project them into local neighborhoods for Graph Flow.

Generation relation. Every admitted child records the single selected factor from which it was mutated. The stored provenance is directed, but Graph Flow uses a symmetric unit-weight edge between each direct parent–child pair. No explicit edges are added between more distant ancestors and descendants; their influence arises through repeated propagation. Invalid or duplicate proposals do not create generation edges.

Structure relation. For each factor, CosmosAlpha extracts the set of canonical subexpressions from its AST. A subexpression preserves its operator, child topology, primitive fields, constants, and window arguments, and commutative operands are ordered before extraction. Structural similarity is the Jaccard similarity in Eq. 3; because the representation is a set, repeated occurrences of an identical subtree are counted once.

Correlation relation. Correlation is calculated from the evaluated and preprocessed stock exposures, not from raw expression outputs. For a factor pair, the calculation first requires the two

exposure histories to overlap on at least 80% of the evaluation calendar. On each overlapping date, Pearson correlation is computed over stocks for which both exposures are finite and eligible. A daily value is retained only if this intersection contains at least 100 stocks and covers at least 80% of that day's eligible universe. The remaining daily correlations are averaged with equal date weights as in Eq. 4. A pair can form an edge only when valid daily correlations cover at least 80% of the full evaluation calendar and at least 80% of its overlapping dates. The signed value C_{ij} is retained as a correlation fact, while relation strength and distance use $|C_{ij}|$ and $1 - |C_{ij}|$, respectively.

Local neighborhood projection. Structure candidates must satisfy $S_{ij}^{\mathrm{str}} \geq 0.20$, whereas reliable correlation candidates must satisfy $|C_{ij}| \geq 0.10$. For each relation separately, a node ranks the remaining candidates by similarity and retains at most its Top-30 neighbors. The final undirected graph uses union symmetrization: edge (i, j) is retained if either i selects j or j selects i, with weight S_{ij}^{str} or $|C_{ij}|$. A node's final undirected degree can therefore exceed 30. Before the three modalities are combined, each is normalized independently; a row with no retained neighbor is replaced by a self-loop.

A.4 Archive-Based Search Utilities

Quality normalization and archive. The Quality Pareto Archive applies dominance to the unclipped raw vector in Eq. 6. Fixed anchors are used only to construct scalar node Quality and to calculate hypervolume. The normalization ranges are $[0, 0.05]$ for absolute mean RankIC, $[0.05, 0.09]$ in the reverse direction for RankIC standard deviation, and $[0, 8 \times 10^{-4}]$ for direction-aligned Top-20% return. The resulting coordinates are

$$\tilde{q}_i^{\mathrm{IC}} = \mathrm{clip}\left(\frac{|\overline{\mathrm{RankIC}}_i|}{0.05}, 0, 1\right), \tag{11}$$

$$\tilde{q}_i^{\mathrm{std}} = \mathrm{clip}\left(\frac{0.09 - \sigma(\mathrm{RankIC}_i)}{0.09 - 0.05}, 0, 1\right), \tag{12}$$

$$\tilde{q}_i^{\mathrm{ret}} = \mathrm{clip}\left(\frac{R_i^{\mathrm{Top20}}}{8 \times 10^{-4}}, 0, 1\right). \tag{13}$$

Node Quality is their equal-weight mean,

$$Q_i^{\mathrm{self}} = \frac{\tilde{q}_i^{\mathrm{IC}} + \tilde{q}_i^{\mathrm{std}} + \tilde{q}_i^{\mathrm{ret}}}{3}. \tag{14}$$

The anchors are fixed before search and are never estimated from a current batch or graph. Hypervolume is calculated from the same normalized coordinates in $[0, 1]^3$ with reference point $(0, 0, 0)$; archive dominance itself remains defined on the raw objectives.

Fertility. At the start of round r, the current Quality Pareto Archive is fixed as $\mathcal{A}_r^-$. Every valid unique child in that round receives the birth-time gain in Eq. 7 relative to this same archive. These gains are computed independently, so neither completion order nor a sibling's quality changes another child's reward. The archive is updated jointly only after all birth-time gains have been assigned.

An expansion attempt may produce more than one valid child. If $\mathcal{C}_e$ is the set of valid unique children from attempt e, its productive outcome is

$$y_e = \begin{cases} \max_{c \in \mathcal{C}_e} \Delta HV_c, & \mathcal{C}_e \neq \emptyset, \\ 0, & \mathcal{C}_e = \emptyset. \end{cases} \tag{15}$$

Thus, failed, invalid, duplicate-only, and otherwise childless attempts all contribute zero. Fertility operationalizes a parent's productive potential as the mean of its three best attempt outcomes,

$$Q_i^{\mathrm{fert}} = \mathrm{Top3Mean}(y_{i,1}, y_{i,2}, \ldots), \tag{16}$$

where missing outcomes are padded with zero when fewer than three attempts have been observed.

Diversity reference set. At the beginning of each round, a high-quality reference set $\mathcal{R}_r$ is selected from the current Quality Pareto Archive. This set is held fixed throughout the round: children generated in round r are compared with $\mathcal{R}_r$ and can enter the reference set only from the following round onward. We use $K = 10$ for both relation spaces. For each relation $m \in \{\mathrm{str}, \mathrm{corr}\}$, the node itself is excluded and the closest available reference factors define

$$D_i^m = \frac{1}{|\mathcal{N}_K^m(i)|} \sum_{j \in \mathcal{N}_K^m(i)} d_{ij}^m, \qquad Q_i^{\mathrm{div}} = \frac{D_i^{\mathrm{str}} + D_i^{\mathrm{corr}}}{2}. \tag{17}$$

If fewer than K references are available for a modality, all available references are used; if none remains after self-exclusion and validity checks, that modality contributes zero. Generation distance is excluded because lineage is already represented directly by the generation graph.

A.5 Graph Flow and Pareto Parent Selection

The relation matrices are row-normalized independently and combined as in Eq. 5. The self-loop assigned to an empty modality row preserves that node's own signal for that modality. In particular, if a node is isolated under all three relations, its combined transition row is a self-loop and Graph Flow leaves its utility equal to its raw utility. Starting from $F^{(0)} = R$, the implementation evaluates Eq. 10 with $\gamma = 0.75$. Fixed-point iteration stops when the maximum absolute change in any utility coordinate is at most 10^{-8} or after 100 iterations.

A node is eligible for expansion while it has made fewer than five previous attempts. Eligible nodes are sorted into non-dominated fronts in the propagated utility space $\mathbf{F}_i = (F_i^{\mathrm{self}}, F_i^{\mathrm{fert}}, F_i^{\mathrm{div}})$. Up to four parents are taken without replacement from successive fronts in each round. If the next front exceeds the remaining slots, standard NSGA-II crowding distance prioritizes less crowded points (Deb et al., 2002); any remaining ties are resolved deterministically. Diversity therefore enters parent selection through the propagated AverageDistance objective and, when a front must be truncated, through crowding in the three-dimensional utility space. The selected batch is the minimum of four, the number of eligible nodes, and the remaining global budget, so the final batch may be smaller than earlier batches.

The global budget counts parent expansion attempts. Each selected parent makes one attempt in the round, and success, duplication, expression invalidity, evaluation failure, and LLM failure all consume one unit. Expression-repair calls within that attempt do not consume additional graph-search budget.

A.6 Parent-Grounded Mutation

Each expansion is conditioned on one selected parent rather than on an undifferentiated collection of prior factors. The mutation context combines the parent expression with its raw quality metrics, current Quality, Fertility, AverageDistance, expansion history, objective-wise gaps to the current archive, and correlation-neighborhood constraints. This context directs the LLM to propose up to two structural or signal-level mutations of the parent instead of unrelated factors.

The resulting designs are converted into expressions under the shared factor language and then pass through validation, optional repair, batch-level duplicate checking, factor calculation, and evaluation. Only valid unique candidates satisfying the node-admission conditions are inserted into the graph. The current implementation uses single-parent mutation; it does not combine two parents through crossover.

A.7 Round-Level Search Procedure

Algorithm 1 makes the round boundaries explicit. The key invariant is that both the Quality Pareto Archive and the high-quality reference set are fixed before parent selection and remain unchanged while all attempts in the batch are evaluated.

Within a batch, deterministic duplicate resolution prevents candidate completion order from deciding which equivalent proposal is retained. Likewise, evaluating all children against $\mathcal{A}_r^-$ makes birth-time hypervolume gains independent of child completion order. At the next round boundary,

Algorithm 1 CosmosAlpha Multi-Relational Graph Search

Require: LLM, common factor evaluator, expansion-attempt budget B
Ensure: Evaluated factor graph $\mathcal{G}$ and Quality Pareto Archive $\mathcal{A}$

```
Generate, validate, deduplicate, and evaluate the initial factors
Admit technically valid factors; construct G and A
while fewer than B expansion attempts have been used do
    Fix the pre-round archive A_r^- and reference set R_r
    Compute AverageDistance to R_r for all current nodes
    Build P; run Graph Flow from the current vector utilities
    Select an eligible parent batch by Pareto fronts and crowding distance
    if no eligible parent can be selected then
        break
    end if
    for each selected parent do
        Generate parent-grounded mutations with the LLM
        Validate candidates and resolve committed and same-batch duplicates
        Calculate and evaluate the surviving candidates
    end for
    Compute every valid child's ΔHV against A_r^-
    Record attempt outcomes and update parent Fertility
    Commit all valid unique children jointly to A
    Insert the children and update generation, structure, and correlation relations
    Compute new-node diversity against the fixed set R_r
end while
return G, A
```

the updated archive determines a new reference set and all node utilities are refreshed before parent selection.

B Metrics and Evaluation Definitions

This appendix defines the quantities used for search-time factor evaluation and for the final out-of-sample comparison. Following the experimental protocol in Appendix C, search-time statistics are computed only on the training period, whereas integrated predictive and portfolio metrics are computed on the held-out test period. All temporal averages below assign equal weight to each valid trading date.

B.1 Single-Factor Metrics

For factor f_i on date t, let $z_{i,j,t}$ denote its processed exposure for stock j, and let $y_{j,t}$ be the corresponding raw one-day forward return. The valid cross-section $\mathcal{U}_{i,t}$ contains eligible stocks for which both quantities are finite, with $n_{i,t} = |\mathcal{U}_{i,t}|$. A date is retained when $n_{i,t} \geq 100$ and the valid observations cover at least 80% of the eligible universe on that date. In addition, a factor admitted as a graph node must have at least two valid dates covering at least 80% of the candidate evaluation dates.

For two cross-sectional vectors a and b over a valid set $\mathcal{U}$, we write their Pearson correlation as

$$\mathrm{Corr}_{\mathcal{U}}(a,b) = \frac{\sum_{j\in\mathcal{U}}(a_j-\bar{a})(b_j-\bar{b})}{\sqrt{\sum_{j\in\mathcal{U}}(a_j-\bar{a})^2}\sqrt{\sum_{j\in\mathcal{U}}(b_j-\bar{b})^2}}. \tag{18}$$

The daily information coefficient and rank information coefficient are

$$\mathrm{IC}_{i,t} = \mathrm{Corr}_{\mathcal{U}_{i,t}}(z_{i,\cdot,t}, y_{\cdot,t}), \tag{19}$$

$$\mathrm{RankIC}_{i,t} = \mathrm{Corr}_{\mathcal{U}_{i,t}}\big(\mathrm{rank}(z_{i,\cdot,t}), \mathrm{rank}(y_{\cdot,t})\big), \tag{20}$$

where tied observations receive their average rank. Dates with a degenerate cross-section, for which either correlation is undefined, are excluded.

Let $\mathcal{T}_i$ be the factor's valid-date set and $T_i = |\mathcal{T}_i|$. For $x_{i,t} \in \{\mathrm{IC}_{i,t}, \mathrm{RankIC}_{i,t}\}$, its time-series mean, sample standard deviation, and information ratio are

$$\bar{x}_i = \frac{1}{T_i} \sum_{t \in \mathcal{T}_i} x_{i,t}, \tag{21}$$

$$s(x_i) = \sqrt{\frac{1}{T_i - 1} \sum_{t \in \mathcal{T}_i} (x_{i,t} - \bar{x}_i)^2}, \tag{22}$$

$$\mathrm{IR}(x_i) = \frac{\bar{x}_i}{s(x_i)}. \tag{23}$$

The temporal means retain their signs; the absolute value is introduced only when mean RankIC enters the Quality vector in Eq. 6. Accordingly, ICIR and RankICIR are the unannualized ratios obtained from the daily IC and RankIC series. The search additionally uses $s(\mathrm{RankIC}_i)$ as its RankIC standard deviation. If fewer than two valid dates are available, or if the sample standard deviation is numerically zero, the corresponding ratio is set to zero; the graph-admission rule separately excludes factors without two valid dates.

The Top-20% return aligns each factor before forming the upper-exposure group. Its orientation is fixed from the factor's mean RankIC,

$$o_i = \begin{cases} +1, & \overline{\mathrm{RankIC}}_i \geq 0, \\ -1, & \overline{\mathrm{RankIC}}_i < 0. \end{cases} \tag{24}$$

Let $k_{i,t} = \lfloor 0.2 n_{i,t} \rfloor$ and let $\mathcal{H}_{i,t}$ contain the $k_{i,t}$ stocks with the largest values of $o_i z_{i,j,t}$. Exposure ties at the selection boundary are resolved by instrument identifier. The daily and time-averaged returns are

$$r_{i,t}^{\mathrm{Top20}} = \frac{1}{k_{i,t}} \sum_{j \in \mathcal{H}_{i,t}} y_{j,t}, \qquad R_i^{\mathrm{Top20}} = \frac{1}{T_i} \sum_{t \in \mathcal{T}_i} r_{i,t}^{\mathrm{Top20}}. \tag{25}$$

This quantity is a search-time single-factor statistic based on raw forward returns; it is distinct from the downstream TopK-Drop portfolio backtest.

B.2 Integrated Predictive Metrics

The final predictive metrics use the same daily definitions but evaluate the integrated LightGBM signal rather than an individual factor. Specifically, the factor exposure in Eqs. 19 and 20 is replaced by the model prediction $\hat{y}_{j,t}$, while the correlation target remains the raw one-day forward return $y_{j,t}$. A test date is valid when at least 100 stocks have finite predictions and returns. The reported IC and RankIC are the signed means of the valid daily correlations; ICIR and RankICIR divide those means by their sample standard deviations as in Eq. 23, without annualization or taking absolute values.

B.3 Portfolio Metrics

The LightGBM signal is converted into daily holdings by the Qlib TopK-Drop backtest described in Appendix C. Let r_t be its daily absolute portfolio return; neither a market benchmark nor a risk-free rate is subtracted. Consistent with Qlib's daily arithmetic-return analysis (Yang et al., 2020), let $A = 238$, and define

$$\bar{r} = \frac{1}{T} \sum_{t=1}^{T} r_t, \qquad s_r = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} (r_t - \bar{r})^2}. \tag{26}$$

Annualized return and portfolio information ratio are then

$$\mathrm{ARR} = A\bar{r}, \qquad \mathrm{IR} = \sqrt{A}\, \frac{\bar{r}}{s_r}. \tag{27}$$

For signed maximum drawdown, define the arithmetic cumulative return $C_t = \sum_{u=1}^{t} r_u$. We report

$$\mathrm{MDD} = \min_{1 \leq t \leq T} \left(C_t - \max_{1 \leq u \leq t} C_u \right) \leq 0. \tag{28}$$

Thus, a larger signed MDD (closer to zero) indicates a smaller drawdown. ARR and MDD are multiplied by 100 when displayed as percentages in the tables.

For method m, the final Mean Rank summarizes the four predictive and three portfolio metrics:

$$\mathrm{MeanRank}_m = \frac{1}{7}\sum_{q=1}^{7} \mathrm{rank}_{m,q}\,. \tag{29}$$

Rank 1 is best and ties receive their average rank. All seven columns are ranked in descending order because MDD is signed; hence a smaller Mean Rank is better.

B.4 Repeated-Run Aggregation

The stability analysis uses $L = 3$ independent LLM runs under the same data, model, and search configuration. For a reported metric value v_ℓ from run ℓ, the Mean and Std. rows are

$$\bar{v} = \frac{1}{L}\sum_{\ell=1}^{L} v_\ell, \qquad \sigma_{\mathrm{run}} = \sqrt{\frac{1}{L}\sum_{\ell=1}^{L}(v_\ell - \bar{v})^2}. \tag{30}$$

The reported run-level dispersion therefore uses the population standard deviation across the three observed runs. It should not be confused with the sample standard deviation across trading dates used by ICIR, RankICIR, and portfolio IR.

C Experimental Configuration

This appendix consolidates the principal data, search, model, and backtest settings used in the reported experiments. Unless a row is explicitly marked as Graph-specific or ablation-only, the CSI 300 and CSI 500 experiments use the same setting. Metric formulas are defined separately in Appendix B.

C.1 Data, Temporal Splits, and Common Evaluation

All methods are evaluated on the same daily A-share snapshot. Historical index membership and trading-eligibility masks are applied point-in-time. Factor mining and all search feedback use only the training period; the mined factor pool is then fixed before downstream validation and testing. Table 4 summarizes the common data and factor-evaluation protocol.

The downstream target is the same one-day return, cross-sectionally rank normalized for LightGBM training. Predictions are evaluated against the raw forward return as specified in Appendix B.2. No additional pairwise-correlation filter is applied before model fitting.

C.2 Search Budget and Expression Generation

Each CosmosAlpha run begins with factors generated within that run. Automated methods are budget-matched to produce approximately 150 evaluable factors for downstream integration. Because the search organizations consume proposals differently, this common output target is used for comparison rather than equating an MCTS iteration, a Chain step, and a Graph expansion attempt. Downstream integration uses the complete resulting factor pool.

An expansion attempt remains part of the search budget when it yields an invalid or duplicate proposal, as detailed in Appendix A.2. Prompt text is provided in Appendix D; the complete factor language is specified in Appendix E.

C.3 Graph Construction and Search Hyperparameters

Tables 6 and 7 collect the parameters that instantiate the Graph method described in Section 3. Relation thresholds are applied before per-node Top-K truncation, and all values are fixed before a run begins.

Table 4: Shared data, preprocessing, and factor-evaluation configuration.

Setting	Value	Scope or provenance
Universes	CSI 300 and CSI 500 with historical membership	Shared across methods
Frequency and prices	Daily; post-adjusted prices	Shared data protocol
Rolling lookback start	January 4, 2010	Supports lagged operators
Mining and training	April 1, 2010–May 31, 2019	Search feedback and LightGBM fitting
Validation	June 1, 2019–December 31, 2020	LightGBM early stopping only
Test	January 1, 2021–December 31, 2025	Final prediction and portfolio evaluation only
Return label	One-trading-day forward close-to-close return	Shared across methods
Stock eligibility	At least one year since listing; blacklist and tradability masks enabled; financial stocks retained	Shared data protocol
Rolling-data requirement	At least two thirds of a window; sample standard deviation uses $\mathrm{ddof} = 1$	Factor calculation
Cross-sectional processing	MAD winsorization with multiplier 5, neutralization, then Z-score normalization	Shared across methods
Neutralization exposures	Industry, log market capitalization, 20-day return, turnover, and volatility	Shared across methods
Minimum daily sample	100 finite eligible stocks and at least 80% daily coverage	Factor evaluation
Minimum date coverage	At least two valid dates covering 80% of candidate evaluation dates	Graph node admission

Table 5: Search and expression-generation settings for CosmosAlpha-Graph.

Setting	Value	Role
Initial Graph factors	2	LLM-generated factors evaluated before the first expansion round
Parents per round	Up to 4	Parallel expansion width
Expansion cap per node	5 attempts	Prevents repeated use of one parent
Child proposals per attempt	Up to 2	Parent-grounded mutations
Factor-pool target	Approximately 150 evaluable factors	Common downstream budget across automated methods
Maximum expression depth	15	Expression-complexity constraint
Allowed rolling windows	$\{5, 10, 20\}$ trading days	Shared factor language
Expression-repair attempts	At most 2	Applied after an invalid generated expression
Duplicate policy	Reject an existing canonical expression	Shared factor-identity rule

Table 6: Graph relation and neighborhood settings.

Setting	Value	Role
Structure edge threshold	0.20	Minimum Jaccard similarity
Structure neighbors	Top-30 per node	Local structure neighborhood before union symmetrization
Correlation edge threshold	$\lvert C_{ij}\rvert \geq 0.10$	Minimum exposure-correlation strength
Correlation neighbors	Top-30 per node	Local correlation neighborhood before union symmetrization
Correlation daily validity	100 stocks and 80% coverage	Reliable daily cross-sectional correlation
Correlation date validity	80% of the evaluation calendar and 80% of overlapping dates	Reliable pairwise relation
AverageDistance neighbors	$K = 10$ per relation	Nearest high-quality references in structure and correlation spaces
Generation edge weight	1	Direct parent–child connection

Table 7: Quality, Fertility, Graph Flow, and parent-selection settings.

Setting	Value	Role
$\lvert\overline{\mathrm{RankIC}}\rvert$ anchors	$[0, 0.05]$	Quality and hypervolume normalization
RankIC standard-deviation anchors	$[0.05, 0.09]$, reversed	Quality and hypervolume normalization
Top-20% return anchors	$[0, 8 \times 10^{-4}]$	Quality and hypervolume normalization
Hypervolume reference point	$(0, 0, 0)$	Quality Pareto Archive
Fertility history	Top-3 attempt outcomes	Productive-potential aggregation
Relation weights	$1/3$ each	Equal generation, structure, and correlation contribution
Graph Flow coefficient	$\gamma = 0.75$	Neighbor-propagation strength
Graph Flow solver	At most 100 iterations; tolerance 10^{-8}	Fixed-point convergence
Parent selection	Pareto fronts with crowding distance	Multi-objective selection
Crowding tolerance	10^{-12}	Degenerate-objective handling

C.4 LLM CONFIGURATION

DeepSeek-V4-Flash is the default backbone for the main comparison. The prompt renderer permits at most 32,000 input tokens, retains up to four history items, and requests at most 12,000 output tokens. At most two formatting retries are allowed. Table 8 lists the model-side settings; the GPT and Kimi rows apply only to the sensitivity study in Appendix I.

Table 8: LLM settings used in the main and backbone-sensitivity experiments.

Use	Model identifier	Temperature	Reasoning control
Main	DeepSeek-V4-Flash	0.0	Maximum
Sensitivity	GPT-5.6-Sol	0.0	xhigh
Sensitivity	Kimi K3	1.0	Maximum

Observed LLM usage. With a budget of 100 Graph expansion attempts, average per-run usage was 6.17 hours of cumulative LLM call time and 3.98M tokens (1.56M input and 2.41M output) for DeepSeek-V4-Flash; 6.87 hours and 1.42M tokens (0.96M input and 0.46M output) for Kimi K3; and 2.18 hours and 1.72M tokens (1.25M input and 0.47M output) for GPT-5.6-Sol. Cumulative call time sums recorded client-call durations rather than end-to-end wall-clock time. Token totals are lower bounds because some terminal records lack provider-reported counts.

C.5 LIGHTGBM AND PORTFOLIO CONFIGURATION

All factor pools are recomputed on the common data snapshot and passed to the same LightGBM and Qlib backtest pipeline. The LightGBM settings follow the Qlib Alpha158 benchmark (Yang et al., 2020). Table 9 reports the principal settings that affect the displayed predictive and portfolio results.

Table 9: Shared downstream LightGBM and portfolio configuration.

Component	Setting	Value
LightGBM	Objective	Mean squared error
LightGBM	Learning rate; boosting cap; early stopping	0.2; 1000 rounds; 50 rounds
LightGBM	Column and row subsampling	0.8879 and 0.8789
LightGBM	L_1 and L_2 penalties	205.6999 and 580.9768
LightGBM	Maximum depth; leaves	8; 210
Portfolio	Initial account; invested fraction	10^8; 0.95
Portfolio	TopK-Drop rule	Hold 50 stocks; replace up to 5 daily
Portfolio	Buy, sell, and minimum-hold rules	Buy highest scores; sell lowest scores; hold threshold 1 day
Portfolio	Execution and price-limit threshold	Open price; 0.095
Portfolio	Trade unit	100 shares
Portfolio	Return convention	Absolute return with a zero benchmark

Predictions formed from information available through trading day t are first used for portfolio decisions at the open of trading day $t + 1$.

The implementation requires Python ≥ 3.12, LightGBM $\geq 4.6, < 5$, and Qlib $\geq 0.9.7, < 0.10$.

C.6 ABLATION-ONLY CONFIGURATION

The ablations share all preceding settings unless the variant explicitly changes a search decision. Table 10 records the numerical settings used only by the two scalarization controls. The Flow

intervals were fixed from Quality, Fertility, and AverageDistance statistics collected across multiple Graph runs, rather than fitted to the reported ablation outcomes.

Table 10: Parameters used only by the scalarization ablations.

Setting	Value	Provenance
Quality-scalar archive capacity	20 factors	Fixed ablation design
Flow Quality interval	$[0.44, 0.71]$	Repeated Graph-run statistics
Flow Fertility interval	$[0, 0.004]$	Repeated Graph-run statistics
Flow AverageDistance interval	$[0.60, 0.86]$	Repeated Graph-run statistics
Flow scalar weights	$1/3$ each	Equal-weight control

C.7 Baseline Implementations

The descriptions and, where available, repository links below document the implementation sources used for each baseline. Each baseline retains its own factor-discovery procedure. The factors in each resulting pool are recomputed on the common data snapshot and evaluated with the shared preprocessing, LightGBM, and backtest protocol; reported values are not copied from the source papers or repositories.

Alpha158. We use the 158 handcrafted Qlib features as a fixed factor pool. Implementation repository: https://github.com/microsoft/qlib.

GP. The GP baseline adapts the genetic-programming implementation in gplearn to the factor-mining task through a task-specific representation and search interface. Implementation repository: https://github.com/trevorstephens/gplearn.

AlphaGen. AlphaGen uses reinforcement learning to generate synergistic collections of formulaic alpha factors (Yu et al., 2023). We include it as a factor-collection baseline and evaluate its resulting pool under the common downstream protocol. Implementation repository: https://github.com/ICT-FinD-Lab/alphagen.

AlphaForge. AlphaForge mines formulaic alpha factors and dynamically combines them into factor collections (Shi et al., 2025). Its resulting pool is evaluated with the same preprocessing, LightGBM integration, and portfolio backtest as the other methods. Implementation repository: https://github.com/dulyhao/alphaforge.

AlphaSAGE. AlphaSAGE performs structure-aware alpha mining with GFlowNets to encourage diverse exploration (Chen et al., 2026). We use it as an additional structure-aware search baseline and apply the common downstream evaluation protocol to its factor pool. Implementation repository: https://github.com/BerkinChen/AlphaSAGE.

AlphaAgent. AlphaAgent organizes factor discovery as a multi-agent workflow for hypothesis generation, factor construction, and iterative feedback. Implementation repository: https://github.com/RndmVariableQ/AlphaAgent/tree/legacy-main.

R&D-Agent. R&D-Agent is a multi-agent framework that jointly coordinates factor research and model development (Li et al., 2025). We include its factor-mining output as a baseline and recompute downstream metrics under the common evaluation protocol. Implementation repository: https://github.com/microsoft/rd-agent.

AlphaPROBE. AlphaPROBE combines retrieval-guided factor proposal with graph-biased evolution. Implementation repository: https://github.com/gta0804/AlphaPROBE.

QuantaAlpha. QuantaAlpha performs LLM-guided evolutionary search over factor trajectories. Implementation repository: https://github.com/QuantaAlpha/QuantaAlpha.

AlphaJungle and CosmosAlpha-MCTS. AlphaJungle (Shi et al., 2026) is an MCTS-based factor-mining method, but we did not identify an official public implementation at the time of submission, so we do not claim a direct reimplementation. We instead use CosmosAlpha-MCTS as a transparent MCTS-based proxy. It maintains multiple subtrees rather than one large tree to encourage greater factor diversity, while sharing the factor language, evaluator, LLM backbone, and downstream protocol with CosmosAlpha-Graph.

D Prompt Templates

The candidate-generation pipeline separates semantic factor design from expression synthesis. This appendix reports the fixed instructions used to create or repair a candidate factor; runtime JSON payloads are represented by placeholders and described separately below. These are paper-facing transcriptions: escaped line breaks are rendered directly, and implementation-only bookkeeping clauses that do not alter candidate construction are omitted.

Table 11 records which template is used by each search mode. Graph initialization uses the common hypothesis and factor-design templates, whereas later Graph expansions use the parent-grounded template. MCTS uses the common hypothesis template followed by its dedicated factor-design template for both root generation and subsequent expansion. Chain uses the common hypothesis and factor-design templates. All three modes share the expression and repair stages.

Table 11: Versioned prompt routes used for candidate generation.

Template	Version	Search stage	Structured output
Hypothesis	2.3.0	Graph initialization and expansion; MCTS root and expansion; Chain	One testable hypothesis
Factor design	2.2.0	Graph initialization; Chain	One or two semantic designs
Graph factor design	2.2.0	Graph expansion	Parent-grounded mutations
MCTS factor design	2.3.0	MCTS root and expansion	Initial designs or parent-grounded refinements
Factor expression	2.1.0	Shared	One DSL expression per design
Expression repair	2.1.0	Shared, when validation fails	One corrected DSL expression

The templates follow the version-2 prompt manifest used by the implementation.

D.1 Runtime Substitution and Output Contracts

The fixed messages do not enumerate a particular run's factors or numerical search state. Instead, the renderer inserts the stage-specific JSON objects in Table 12. This separation keeps the instructions unchanged while allowing the selected parent, measured search state, and available factor language to vary across calls.

Every response is parsed against a closed JSON contract: unrequested fields are rejected. A hypothesis contains `statement`, `rationale`, `mechanism`, `testable_prediction`, `expected_direction`, `required_fields`, `prediction_horizon`, and `assumptions`. A design response contains a `designs` array; each item contains exactly `name`, `description`, `mechanism_mapping`, `signal_definition`, `variables`, `formulation`, `window_choices`, `expected_direction`, and `failure_conditions`. The expression and repair stages return only `proposed_expression`. The maximum array length is supplied at runtime by `generation_constraints.max_designs` and is never greater than two.

D.2 Hypothesis and Common Factor Design Templates

The hypothesis stage asks for one falsifiable daily equity-factor hypothesis without allowing the model to write an expression. It is used by Chain, by both the initialization and expansion calls

Table 12: Runtime placeholders appearing in the prompt templates.

Placeholder	Runtime content
`context_json`	Research goal, market, stock universe, frequency, available fields, search constraints, and bounded prior search summaries. For Graph expansion, the search constraint carries the selected parent's current guidance.
`request_json`	The current hypothesis and stage-specific request. A design request includes the universe, frequency, generation limit, and relevant parent reference. MCTS expansion additionally supplies the parent expression, scores, and target dimension. An expression request contains one semantic design and expression limits; a repair request contains the previous expression and its deterministic validation issues.
`prompt_view_json`	The runtime capability view: admissible raw fields, canonical operators and signatures, supported windows, and the bounded context permitted at that stage.

of Graph, and by MCTS root generation and subsequent expansion. The paper-facing transcription below retains the instructions that determine hypothesis content while omitting implementation-only evidence bookkeeping.

Hypothesis system message.

```
You are the CosmosAlpha Hypothesis Agent. Produce one falsifiable daily equity-factor
     hypothesis from the supplied research context, runtime capability catalog, and bounded
     structured experiment lessons. Treat prior metrics, stage outcomes, and failure codes as
      evidence; treat summarized lessons and next actions as prior interpretation, not new
     evidence. Keep the hypothesis scoped to the configured universe in the research context;
      do not infer or replace it with current or future constituent membership. Use only
     available fields. Do not write a factor expression or invent evidence. Be concise and
     return only the exact JSON object requested.
```

Hypothesis user message.

```
Research context JSON:
${context_json}

Use context_json.universe as the research stock-pool scope. Do not replace it with another
     universe or use current or future constituent information.

Runtime capability and structured lesson view JSON:
${prompt_view_json}

Return exactly: statement, rationale, mechanism, testable_prediction, expected_direction,
     required_fields, prediction_horizon, assumptions. required_fields must contain only
     field names present in prompt_view_json.catalog.fields. Use prompt_view_json.lessons
     when present and start normally when it is empty. prediction_horizon is a string such as
      "1d" and assumptions is an array of strings.
```

The common factor-design stage, used by Chain and Graph initialization, converts one hypothesis into semantic designs. It deliberately prohibits executable syntax so that economic intent and DSL construction remain separate.

Common factor-design system message.

```
You are the CosmosAlpha Factor Design Agent. Decide what quantitative signal to construct,
     but do not write CosmosAlpha DSL or executable code. Every variable name and window must
      come from the supplied runtime capability view. Variables are raw input fields only;
     intermediate names such as rolling means, ratios, masks, or conditional values must not
     appear in variables and belong only in formulation or signal_definition. Produce one or
     two distinct, testable designs concisely and return only the exact JSON object requested
     .
```

Common factor-design user message.

```
Factor design request JSON:
${request_json}

Use request_json.universe as the research stock-pool scope. Do not replace it with another
     universe or use current or future constituent information.

Runtime capability view JSON:
```

```
${prompt_view_json}

Return a designs array containing no more than generation_constraints.max_designs items. Each
     design must contain exactly: name, description, mechanism_mapping, signal_definition,
    variables (objects with name and role), formulation, window_choices (objects with window
     and rationale), expected_direction, failure_conditions. Every variables[].name must
    exactly match one field name in prompt_view_json.catalog.fields; use only its raw
    catalog name and do not list intermediate variables. variables[].role should describe
    the field's input role. Intermediate quantities belong only in formulation or
    signal_definition. Every window_choices[].window must be one of the supported_windows
    shown by the catalog. formulation must be mathematical or natural-language construction
    guidance, not DSL.
```

D.3 Graph Parent-Grounded Mutation Template

For a Graph expansion, the design call receives only the selected parent as its mutation reference. Before this call, the hypothesis stage receives Graph guidance containing the parent's expression, measured utilities, expansion history, archive-relative gaps, and correlation-neighborhood constraints. The resulting hypothesis and the selected parent reference then form the design request described in Appendix A.6.

Graph factor-design system message.

```
You are the CosmosAlpha Graph Factor Design Agent. Produce a parent-grounded mutation of the
    single current parent factor described by the factor-design request JSON. Use that
    parent as the sole design reference. Propose one or two distinct, testable designs that
    make an explicit structural or signal-level change to the parent. Do not write
    CosmosAlpha DSL or executable code. Every variable name and window must come from the
    supplied runtime capability view. Variables are raw input fields only; intermediate
    names such as rolling means, ratios, masks, or conditional values must not appear in
    variables and belong only in formulation or signal_definition. Return only the exact
    JSON object requested.
```

Graph factor-design user message.

```
Graph parent-grounded factor-design request JSON:
${request_json}

Use request_json.universe as the research stock-pool scope. Do not replace it with another
    universe or use current or future constituent information.

Runtime capability view JSON:
${prompt_view_json}

The request must contain only the current parent in existing_factor_refs. Treat that parent
    as the sole mutation reference. Return a designs array containing no more than
    generation_constraints.max_designs items. Each design must contain exactly: name,
    description, mechanism_mapping, signal_definition, variables (objects with name and role
    ), formulation, window_choices (objects with window and rationale), expected_direction,
    failure_conditions. Every design must describe a mutation of the supplied parent rather
    than an unrelated factor. Every variables[].name must exactly match one field name in
    prompt_view_json.catalog.fields; use only its raw catalog name and do not list
    intermediate variables. variables[].role should describe the field's input role.
    Intermediate quantities belong only in formulation or signal_definition. Every
    window_choices[].window must be one of the supported_windows shown by the catalog.
    formulation must be mathematical or natural-language construction guidance, not DSL.
```

D.4 MCTS Factor-Design Template

MCTS uses the common hypothesis template followed by a dedicated factor-design template. At root generation, its parent-specific fields are unset and the hypothesis provides the initial design direction. During subsequent expansion, the request supplies the selected parent expression, its available scores, and the target search dimension; the model is instructed not to infer these quantities from the hypothesis prose.

MCTS factor-design system message.

```
You are the CosmosAlpha MCTS Factor Design Agent. Refine the supplied parent factor rather
    than inventing an unrelated factor. The factor-design request JSON carries the MCTS
    search context: parent_expression (the exact parent factor expression), parent_scores (
    its measured RankIC, Stability, Diversity, and Turnover when present), focus_dimension (
    the single search dimension this refinement must target), plus the parent metrics,
    target dimension, and refinement history or search guidance. Use parent_expression,
    focus_dimension, and parent_scores as the authoritative design basis; never reconstruct
```

```
    or guess them from the hypothesis text. The hypothesis mechanism is only a suggestion:
    if it cannot be expressed with the listed operators (for example, if it uses unsupported
     conditions, thresholds, or transforms), ignore that detail and refine the parent toward
     focus_dimension using only the listed operators. Propose one or two distinct, testable
    designs with an explicit relationship to the parent and a clear improvement objective
    for the target dimension. Do not write CosmosAlpha DSL or executable code. Every
    variable name and window must come from the supplied runtime capability view. Variables
    are raw input fields only; intermediate names such as rolling means, ratios, masks, or
    conditional values must not appear in variables and belong only in formulation or
    signal_definition. Return only the exact JSON object requested.
```

MCTS factor-design user message.

```
MCTS factor-design request JSON:
${request_json}

Use request_json.universe as the research stock-pool scope. Do not replace it with another
    universe or use current or future constituent information.

Runtime capability view JSON:
${prompt_view_json}

Do not enumerate or exhaustively evaluate multiple alternative designs. Pick the single most
    promising refinement and commit to it; output it directly and keep any deliberation
    brief. Return a designs array containing no more than generation_constraints.max_designs
     items. Each design must contain exactly: name, description, mechanism_mapping,
    signal_definition, variables (objects with name and role), formulation, window_choices (
    objects with window and rationale), expected_direction, failure_conditions. Every design
     must refine the supplied parent factor and address the supplied target dimension; do
    not silently replace the parent with an unrelated idea. Every variables[].name must
    exactly match one field name in prompt_view_json.catalog.fields; use only its raw
    catalog name and do not list intermediate variables. variables[].role should describe
    the field's input role. Intermediate quantities belong only in formulation or
    signal_definition. Every window_choices[].window must be one of the supported_windows
    shown by the catalog. formulation must be mathematical or natural-language construction
    guidance, not DSL.
```

D.5 Expression Synthesis and Repair Templates

The expression agent translates one semantic design into the canonical DSL; it is not allowed to redesign the signal. If deterministic validation fails, the repair agent receives the rejected expression and concrete validation issues and may correct only those errors. The numerical expression limits and validity rules are specified in Appendix E; the run-specific depth cap and repair budget are reported in Appendix C.

Factor-expression system message.

```
You are the CosmosAlpha Factor Expression Agent. Translate exactly one supplied FactorDesign
    into one CosmosAlpha DSL expression. Use only fields, canonical operator names,
    signatures, and windows in the runtime capability view. Do not redesign the signal or
    return metadata. Be concise and return only the exact JSON object requested.
```

Factor-expression user message.

```
Factor expression request JSON:
${request_json}

Runtime capability view JSON:
${prompt_view_json}

Return exactly {"proposed_expression": "..."}. The expression must be a field, number, or
    lowercase function call. Do not use infix arithmetic, grouping parentheses, Qlib syntax,
     Python, strings, attributes, keyword arguments, IDs, descriptions, or directions.
```

Expression-repair system message.

```
You are the CosmosAlpha Expression Repair Agent. Correct only the deterministic expression
    errors supplied in the request. Preserve the FactorDesign mechanism, signal definition,
    direction, and formulation. Use only fields, canonical operators, signatures, and
    windows in the runtime capability view. Do not claim validity or change the design. Be
    concise and return only the exact JSON object requested.
```

Expression-repair user message.

```
Expression repair request JSON:
${request_json}
```

```
Runtime capability view JSON:
${prompt_view_json}

The request contains the complete previous expression and deterministic validation issues.
    Repair those errors, then return exactly {"proposed_expression": "..."}. Do not return
    IDs, metadata, explanations, design changes, or validation status.
```

E FACTOR LANGUAGE AND SEARCH SPACE

CosmosAlpha restricts candidate generation to a fixed domain-specific language (DSL). Versioned field and operator catalogs define the capabilities supplied to the LLM and accepted by the validator; a deterministic factor calculator then executes each accepted expression. This section defines the resulting search space: the admissible expression fields, the complete operator set, and the syntactic and complexity constraints imposed before evaluation.

E.1 FIELD CATALOG

Only the nine daily market fields in Table 13 may appear as terminals in a factor expression. Prices are post-adjusted, turnover and returns are represented as dimensionless decimal values, and no additional clipping or rescaling is performed by the expression engine. Missing source observations are represented explicitly and handled according to the operator rules below.

Table 13: Fields admissible in factor expressions. All fields are observed at daily frequency. The domain column describes the natural data domain; entries may also be missing.

DSL field	Symbol	Definition	Scale or natural domain
`open`	$O_{i,t}$	Opening price on trading day t	Post-adjusted price; positive on eligible rows
`high`	$H_{i,t}$	Intraday highest price	Post-adjusted price; positive on eligible rows
`low`	$L_{i,t}$	Intraday lowest price	Post-adjusted price; positive on eligible rows
`close`	$P_{i,t}$	Closing price	Post-adjusted price; positive on eligible rows
`volume`	$V_{i,t}$	Daily trading volume	Source volume unit; nonnegative
`vwap`	$P^{\mathrm{VWAP}}_{i,t}$	Volume-weighted average price	Same adjusted-price scale
`turn`	$\tau_{i,t}$	Turnover relative to total shares outstanding	Decimal ratio; nonnegative
`free_turn`	$\tau^{\mathrm{free}}_{i,t}$	Turnover relative to free-float shares	Decimal ratio; nonnegative
`returns`	$r_{i,t}$	Daily return supplied by the data source	Decimal return; real-valued

Table 14 separates variables needed by the calculation and evaluation pipeline from terminals that the LLM may select. These variables therefore cannot be inserted directly into a generated expression. The industry code is consumed internally by `indneutralize`; all five context variables are also used by the common cross-sectional preprocessing protocol in Table 4. The forward return is an evaluation target only, as formalized in Appendix B.1, and is never exposed as a factor input.

E.2 OPERATOR CATALOG

The DSL contains 26 canonical operator names: 13 non-windowed and 13 time-series operators. Every operator maps one or more aligned stock–date scalar series to another stock–date scalar series. The three supported windows yield 52 instantiated primitives—13 non-windowed and 13×3 windowed variants—before recursive composition. In Table 15, "Comm." marks operators whose inputs are reordered only when CosmosAlpha constructs Graph identities and structural representations; ordinary expression serialization preserves the written argument order.

Table 14: Context and target variables excluded from factor expressions. All variables are aligned daily and constructed point-in-time.

Variable	Symbol	Definition	Role and scale
`industry_id`	$g_{i,t}$	Point-in-time industry classification	Categorical operator and preprocessing context
`log_mkt_cap`	$m_{i,t}$	Natural logarithm of point-in-time market capitalization	Real-valued preprocessing exposure
`ret20`	$r_{i,t}^{(20)}$	Close-to-close return over the trailing 20 trading days through t	Decimal preprocessing exposure
`turn20`	$\bar{\tau}_{i,t}^{(20)}$	Mean daily turnover over the trailing 20 trading days	Nonnegative preprocessing exposure
`vol20`	$\sigma_{i,t}^{(20)}$	Sample standard deviation of daily returns over the trailing 20 trading days	Nonnegative preprocessing exposure
Forward return	$y_{i,t}$	One-trading-day forward close-to-close return	Real-valued prediction target

Table 15: Complete operator catalog. Here rank_t is the same-day cross-sectional percentile rank over eligible finite stocks, with average ranks for ties; $\varepsilon = 10^{-12}$; and every window argument is an integer $w \in \{5, 10, 20\}$ measured in trading days.

Signature	Scope	Definition and numerical behavior	Comm.
Non-windowed operators			
`add(x, y)`	Elementwise	$x + y$ for finite inputs	Yes
`sub(x, y)`	Elementwise	$x - y$ for finite inputs	No
`mul(x, y)`	Elementwise	xy for finite inputs	Yes
`div(x, y)`	Elementwise	x/y; missing when $\lvert y\rvert \le \varepsilon$	No
`abs(x)`	Elementwise	$\lvert x\rvert$	No
`sqrt(x)`	Elementwise	$\sqrt{x}$ for $x \ge 0$; otherwise missing	No
`log(x)`	Elementwise	$\log x$ for $x > 0$; otherwise missing	No
`inv(x)`	Elementwise	$1/x$; missing when $\lvert x\rvert \le \varepsilon$	No
`rank(x)`	Cross-sectional	$\mathrm{rank}_t(x) \in (0, 1]$	No
`rank_sub(x, y)`	Cross-sectional	$\mathrm{rank}_t(x) - \mathrm{rank}_t(y)$	No
`rank_div(x, y)`	Cross-sectional	$\mathrm{rank}_t(x)/\mathrm{rank}_t(y)$ with protected division	No
`sigmoid(x)`	Elementwise	$(1 + \exp(-x))^{-1}$	No
`indneutralize(x)`	Industry group	x minus its same-day mean over eligible finite stocks in industry $g_{i,t}$	No
Time-series operators			
`delay(x, w)`	Time series	Value $x_{i,t-w}$ from exactly w trading rows earlier	No
`delta(x, w)`	Time series	$x_{i,t} - x_{i,t-w}$	No
`corr(x, y, w)`	Time series	Rolling Pearson correlation on pairwise finite observations	Yes
`cov(x, y, w)`	Time series	Rolling sample covariance on pairwise finite observations	Yes
`decay_linear(x, w)`	Time series	Linearly weighted mean with weights $1, \ldots, w$, largest for the most recent observation	No
`ts_min(x, w)`	Time series	Rolling minimum over finite observations	No
`ts_max(x, w)`	Time series	Rolling maximum over finite observations	No
`ts_argmin(x, w)`	Time series	One-based position of the first rolling minimum within the window	No
`ts_argmax(x, w)`	Time series	One-based position of the first rolling maximum within the window	No
`ts_rank(x, w)`	Time series	Percentile rank of $x_{i,t}$ among finite window observations, with averaged ties	No
`ts_sum(x, w)`	Time series	Rolling sum over finite observations	No
`ts_stddev(x, w)`	Time series	Rolling sample standard deviation ($\mathrm{ddof} = 1$)	No
`ts_zscore(x, w)`	Time series	$(x_{i,t} - \bar{x}_{i,t,w})/s_{i,t,w}$ using the rolling sample deviation	No

Elementwise operators propagate missing inputs. Window reducers require at least $\lceil 2w/3 \rceil$ finite observations, while binary rolling statistics apply the same rule to pairwise finite observations; `delay` and `delta` instead require the relevant current and lagged values. The linear-decay weights are renormalized over the available observations. Rolling standard deviations and z-scores are missing when their denominator is below ε, as are correlations with a degenerate input. Any remaining non-finite result is stored as missing before evaluation.

E.3 Grammar, Validation, and Complexity Bounds

An expression is a field, a numeric constant, or a prefix function call whose arguments are themselves expressions:

$$e ::= f \mid c \mid o(e_1, \ldots, e_k) \mid o_w(e_1, \ldots, e_k, w), \qquad f \in \mathcal{F}, \quad w \in \{5, 10, 20\}, \tag{31}$$

where $\mathcal{F}$ is the expression-field set in Table 13. Function and field identifiers are lowercase, and numeric literals use ordinary integer or decimal notation. Arithmetic must use the registered prefix functions; infix operators, standalone grouping parentheses, scientific notation, free-form Python, Qlib syntax, strings, attributes, and keyword arguments are outside the language. Constants may occur within an expression, but every complete factor must reference at least one admissible field.

Table 16 summarizes the hard checks applied before factor calculation. Operators may be nested without a separate operator-pair restriction, subject to the AST-depth and node-count caps. The validator also checks each function's exact arity, its final window literal, the availability of any required context, the daily frequency, and backend support. Consequently, a syntactically valid expression can still be rejected when it lies outside the registered search space.

Table 16: Expression-space and validation constraints. Method-specific search budgets and repair limits are reported in Appendix C.

Constraint	Enforced rule
Data frequency	Daily (`1d`) for every field and operator
Allowed windows	Final integer literal in $\{5, 10, 20\}$ trading days
Expression length	At most 512 characters at the parser boundary
AST depth	Root counted at depth 1; the run-specific cap appears in Table 5
Operator count	At most 32 call nodes
Field count	At most 6 distinct expression fields and at least 1 field
Temporal access	Only contemporaneous or backward-looking fields and windows; target and future-return fields are unavailable
Catalog compliance	Registered lowercase fields and operators, exact arity, available context, and supported backend
Missing values	Undefined domains, insufficient rolling observations, and non-finite outputs remain missing

The syntax normalizer produces deterministic lowercase prefix text but does not perform algebraic simplification or reorder operands. Graph identity and structural extraction then apply the narrower commutative rule indicated in Table 15: the two inputs of `add`, `mul`, `corr`, and `cov` are sorted, while the window of a rolling operator remains fixed. Duplicate handling after this normalization is described in Appendix A.2.

F Qualitative Factor Examples

This section makes the output of CosmosAlpha concrete through four factors from the finalized CSI 300 Graph run. The examples are descriptive rather than an additional performance comparison: each case reports its exact DSL expression, plain-language operation, cautious economic interpretation, and training-period single-factor statistics. E1–E4 are case labels; their six-character factor IDs match the corresponding nodes in the following lineage figure.

F.1 Case Selection and Reporting Protocol

We select cases only from technically valid, unique factors in the final Quality Pareto Archive and do not inspect validation- or test-period outcomes. E1 is the factor with the largest absolute mean RankIC; E2 is the remaining factor with the smallest RankIC standard deviation; and E3 is the remaining factor with the largest direction-aligned Top-20% return. E4 is the remaining archive factor with the largest final-round AverageDistance. If one factor occupies more than one quality extreme, the next distinct factor under the corresponding ordering is used. This rule represents the three intrinsic-quality objectives and a comparatively non-redundant point, rather than selecting four cases by one scalar score or by held-out performance.

Table 17 records each case's graph ID and generation round and depth. The first three statistics follow Appendix B.1; the Top-20 return is direction-aligned and shown in basis points per trading

day. AverageDistance is the final-round value defined in Eq. 8. These are search-time diagnostics on the training split, not the integrated LightGBM test metrics in Tables 1 and 2.

Table 17: Training-period diagnostics for the representative CSI 300 Graph factors. Direction-aligned Top-20 return is reported in basis points (bp) per trading day.

Case	Graph ID	Round/d	$\lvert\overline{\mathrm{RankIC}}\rvert$ ↑	$\sigma(\mathrm{RankIC})$ ↓	Aligned Top-20 (bp/day) ↑	AverageDistance ↑
E1	`63d83d`	18/5	0.058165	0.073829	13.10	0.5092
E2	`835f66`	20/7	0.005304	0.057119	4.93	0.8182
E3	`1b4d31`	21/4	0.054396	0.072582	12.79	0.5040
E4	`5a02af`	19/7	0.017627	0.063906	3.85	0.8535

F.2 Representative Factors

E1: RankIC-oriented case. *Expression:* `decay_linear(div(sub(close,vwap),sub(high,low)),5)`. *Reading:* For each stock, the expression scales the close–VWAP gap by the day's high–low range, then applies a five-day linearly weighted mean that gives more weight to recent observations. *Interpretation:* Its signed training mean RankIC is -0.0582; the direction-aligned Top-20 statistic therefore reverses the raw factor orientation as specified in Appendix B.1. The expression can be read as a smoothed, range-normalized displacement from VWAP, while the observed association alone does not establish an economic mechanism.

E2: stability-oriented case. *Expression:* `rank_sub(ts_zscore(free_turn,5),ts_zscore(div(sub(high,low),open),5))`. *Reading:* It computes five-day time-series z-scores for free-float turnover and the high–low range relative to the opening price, then subtracts their same-day cross-sectional ranks. *Interpretation:* E2 has the smallest RankIC standard deviation among the remaining archive cases (0.0571), although its absolute mean RankIC is only 0.0053. It illustrates the stability criterion; the low variation does not by itself imply a strong mean cross-sectional association.

E3: Top-20-return-oriented case. *Expression:* `mul(decay_linear(div(sub(close,vwap),sub(high,low)),5),rank(free_turn))`. *Reading:* This multiplies the five-day smoothed, range-normalized close–VWAP gap by the same-day cross-sectional rank of free-float turnover. *Interpretation:* Among the remaining archive cases, it has the largest direction-aligned Top-20 training return (12.79 bp per trading day). This is a single-factor search diagnostic and does not establish downstream portfolio performance or explain the source of the association.

E4: diversity-oriented case. *Expression:* `mul(div(close,open),sub(ts_rank(free_turn,10),delay(ts_rank(free_turn,10),5)))`. *Reading:* It multiplies the close-to-open price ratio by the change in the ten-day time-series rank of free-float turnover relative to five trading days earlier. *Interpretation:* E4 has the largest final-round AverageDistance among the remaining archive cases (0.8535), indicating separation under the paper's structure and cross-sectional exposure-correlation distances. This is a diversity diagnostic, not evidence of predictive strength or an economic mechanism.

Taken together, the four cases illustrate distinct selection criteria in the intrinsic Quality Pareto Archive. The next subsection lists all factors in the terminal search Pareto front; the subsequent lineage figure shows their generation paths and front assignments.

F.3 Members of the First Search Pareto Front

The catalog in Table 18 lists all 44 nodes in the terminal first search Pareto front, ordered by generation round and graph ID. This front is formed by non-dominated sorting of propagated Quality, Fertility, and Diversity utilities. It differs from the intrinsic Quality Pareto Archive used to select E1–E4, although seven nodes belong to both sets. The six-character IDs match those in the lineage figure; d denotes generation depth. The two initial factors are marked *Init.* and have generation depth $d = 0$.

Table 18: Canonical expressions for all 44 factors in terminal search Pareto front 1. The ID is the six-character prefix used in the lineage figure; Round is the generation round and d is generation depth.

ID	Round/d	Canonical DSL expression
a6f90c	Init./0	`mul(mul(sub(close,low),inv(sub(high,low))),ts_rank(volume,20))`
ed54a5	Init./0	`rank(mul(sub(div(close,vwap),1),div(turn,div(ts_sum(turn,20),20))))`
65260c	0/1	`mul(div(turn,div(ts_sum(turn,20),20)),div(sub(vwap,close),sub(high,low)))`
4060a1	1/1	`rank(mul(div(sub(close,vwap),sub(high,low)),ts_rank(volume,20)))`
19f0cc	2/3	`mul(div(sub(close,open),open),ts_zscore(free_turn,20))`
f71706	2/2	`mul(ts_zscore(div(sub(close,vwap),sub(high,low)),10),div(volume,div(ts_sum(volume,20),20)))`

Continued on the next page

Table 18: Canonical expressions for terminal search Pareto front 1 (continued).

ID	Round/d	Canonical DSL expression
4b967c	3/1	mul(div(sub(close,low),sub(high,low)),div(sub(turn,div(ts_sum(turn,20),20)),ts_stddev(turn,20)))
c192ad	3/1	mul(div(sub(close,vwap),vwap),ts_zscore(free_turn,20))
cc0000	3/4	mul(sub(div(close,vwap),1),ts_rank(free_turn,20))
fa679f	3/1	rank(mul(rank(div(sub(vwap,low),sub(high,low))),rank(div(free_turn,div(ts_sum(free_turn,20),20)))))
dcbd43	4/2	mul(delta(div(sub(close,vwap),sub(high,low)),5),ts_rank(volume,20))
0cdf59	5/1	mul(rank(div(sub(close,vwap),sub(high,low))),rank(sigmoid(ts_zscore(turn,20))))
a2fda7	5/2	mul(sub(0,rank(sub(div(close,vwap),1))),rank(ts_zscore(free_turn,5)))
c9d72b	5/2	mul(sub(0,rank(ts_zscore(sub(div(close,vwap),1),10))),rank(free_turn))
fc4167	5/1	mul(rank(sub(div(close,vwap),1)),rank(sigmoid(ts_zscore(free_turn,20))))
09a17d	6/4	rank(mul(div(sub(vwap,low),sub(high,low)),corr(div(sub(vwap,low),sub(high,low)),ts_zscore(turn,20),5)))
8419c4	6/4	mul(div(sub(close,vwap),ts_stddev(close,20)),log(add(1,div(volume,decay_linear(volume,10)))))
c4ae40	6/2	mul(div(sub(close,vwap),sub(ts_max(high,10),ts_min(low,10))),ts_rank(volume,10))
f369c2	6/2	mul(ts_sum(div(sub(close,vwap),sub(high,low)),5),ts_rank(volume,5))
35f75c	7/1	mul(sub(turn,delay(turn,5)),div(sub(close,vwap),vwap))
5ecd5e	8/1	rank(mul(sub(div(close,vwap),1),div(mul(turn,20),ts_sum(turn,20))))
6cb762	8/1	rank(mul(mul(sub(div(close,vwap),1),div(turn,div(ts_sum(turn,20),20))),add(1,div(sub(high,low),close))))
1340e2	10/3	decay_linear(mul(free_turn,div(sub(close,vwap),sub(high,low))),5)
3895f5	13/4	mul(div(sub(sub(mul(2,close),high),low),sub(high,low)),corr(div(sub(close,low),sub(high,low)),free_turn,10))
1185be	14/4	decay_linear(mul(rank(free_turn),ts_zscore(div(sub(close,vwap),sub(high,low)),20)),10)
85f0b6	14/4	decay_linear(mul(rank(free_turn),rank(div(sub(close,vwap),sub(high,low)))),5)
3ae17f	17/5	mul(div(div(sub(close,vwap),vwap),ts_stddev(div(sub(close,vwap),vwap),10)),turn)
427d2c	17/5	rank(mul(ts_rank(free_turn,10),rank(sub(close,open))))
fa88b4	17/5	mul(div(sub(close,vwap),sub(high,low)),turn)
63d83d	18/5	decay_linear(div(sub(close,vwap),sub(high,low)),5)
c81478	18/6	mul(rank(div(close,open)),delta(ts_rank(free_turn,10),5))
12e0bc	19/3	mul(ts_zscore(sub(close,vwap),20),ts_rank(volume,10))
5a02af	19/7	mul(div(close,open),sub(ts_rank(free_turn,10),delay(ts_rank(free_turn,10),5)))
751385	19/5	add(rank(div(sub(vwap,low),sub(high,low))),rank(cov(div(sub(vwap,low),sub(high,low)),ts_zscore(turn,20),10)))
7aabfd	19/5	add(rank(div(sub(vwap,low),sub(high,low))),rank(corr(div(sub(vwap,low),sub(high,low)),delta(ts_zscore(free_turn,20),5),5)))
bb8068	19/3	mul(div(sub(close,vwap),sub(ts_max(high,10),ts_min(low,10))),rank(volume))
5b9526	20/5	mul(mul(div(div(sub(vwap,close),sub(high,low)),abs(div(sub(vwap,close),sub(high,low)))),rank(abs(div(sub(vwap,close),sub(high,low))))),rank(turn))
902232	20/8	mul(log(div(close,open)),sub(ts_rank(free_turn,10),delay(ts_rank(free_turn,10),5)))
d819dd	20/8	mul(sub(div(close,open),1),delta(ts_zscore(free_turn,10),5))
8eb196	22/9	mul(rank(sub(div(close,open),1)),rank(delta(free_turn,5)))
5cc5bf	23/3	rank(mul(ts_zscore(div(sub(vwap,close),sub(high,low)),10),div(volume,delay(volume,5))))
88860d	23/3	rank(mul(div(sub(vwap,close),sub(high,low)),div(free_turn,delay(free_turn,5))))
88e162	24/2	mul(sub(div(close,vwap),1),ts_rank(turn,20))
a5dfab	24/9	mul(div(sub(close,vwap),vwap),ts_rank(free_turn,10))

The catalog provides the expressions for the full first front, while the four cases above give more detailed readings of selected archive members. The following lineage figure places both sets within the same generation graph.

G Factor Lineage and Search Pareto Fronts

Figure 3 shows the generation lineage for the CSI 300 Graph run used in Appendix F. Each node's fill color identifies its terminal search Pareto front, computed from the propagated Quality, Fertility, and Diversity utilities. Front 1 contains the non-dominated nodes; subsequent fronts are obtained by repeatedly removing the preceding front before sorting the remaining nodes. Thus, the fill colors encode a node's rank in the three-objective search ordering, not its Quality alone. Counts beside F1–F7 in the legend give the number of nodes in each front.

Edges connect generated factors to their parents. Node labels show a shortened factor ID, generation depth (d), and front index.

H Ablation Details

The ablation variants follow the common CSI 300 data split, LLM configuration, prompt protocol, search budget, and downstream LightGBM evaluation. The two scalarization variants modify separate decisions in the search process.

Intrinsic-quality scalarization. The control keeps the intrinsic-quality Pareto archive and computes the direction-aligned quality coordinates

$$q_i = \left(|\overline{\mathrm{RankIC}}_i|, -\sigma(\mathrm{RankIC}_i), R_i^{\mathrm{Top20}} \right).$$

Quality scalarization retains the fixed normalization in Appendix A.4 and averages its three coordinates:

$$S_i = \frac{\widetilde{q}_i^{\mathrm{IC}} + \widetilde{q}_i^{\mathrm{std}} + \widetilde{q}_i^{\mathrm{ret}}}{3}. \tag{32}$$

This score is also used as the node's SelfQuality in the full model. The ablation additionally retains the highest-scoring factors up to the fixed archive capacity in Table 10, instead of retaining the Pareto first front. Search Pareto selection remains in place. This archive change affects two subsequent quantities.

Quality reference set. AverageDistance measures structure and correlation distance to a high-quality factor set from the current run. In the full model, this set is the intrinsic-quality Pareto archive. Under Quality scalarization, the same role is filled by the highest-scoring factors under S_i. Thus, the ablation changes which factors provide the structure and correlation references, while the pairwise-correlation formula and graph-edge construction remain unchanged. The reference set is fixed for each round and shared by all nodes in that round, and is built only from the current run's factor archive.

Fertility. In the full model, a child's contribution is its increment to the intrinsic-quality archive's hypervolume. Quality scalarization instead measures its gain over the best scalar score in the archive frozen before the round:

$$\Delta S(c) = \max\left(0, S(c) - \max_{a \in \mathcal{A}_r^-} S(a)\right), \tag{33}$$

where $\mathcal{A}_r^-$ is the pre-round scalar archive. Each child is evaluated against that same archive, so siblings do not affect one another's gain. An expansion attempt takes the largest gain among its valid children, and parent Fertility remains the Top-3 mean of historical attempt outcomes. The scalar archive also supplies the factor-quality gaps passed to the generation agent.

Flow scalarization. Flow scalarization retains the intrinsic-quality Pareto archive, hypervolume gain, and Graph Flow propagation. It replaces Pareto front and crowding-based parent ranking with an equal-weight score over the propagated utilities (Q, F, D). For utility $m \in \{Q, F, D\}$, let (ℓ_m, u_m) be its fixed normalization interval from Table 10. The ablation uses

$$z_{m,i} = \mathrm{clip}\left(\frac{F_{m,i} - \ell_m}{u_m - \ell_m}, 0, 1\right), \tag{34}$$

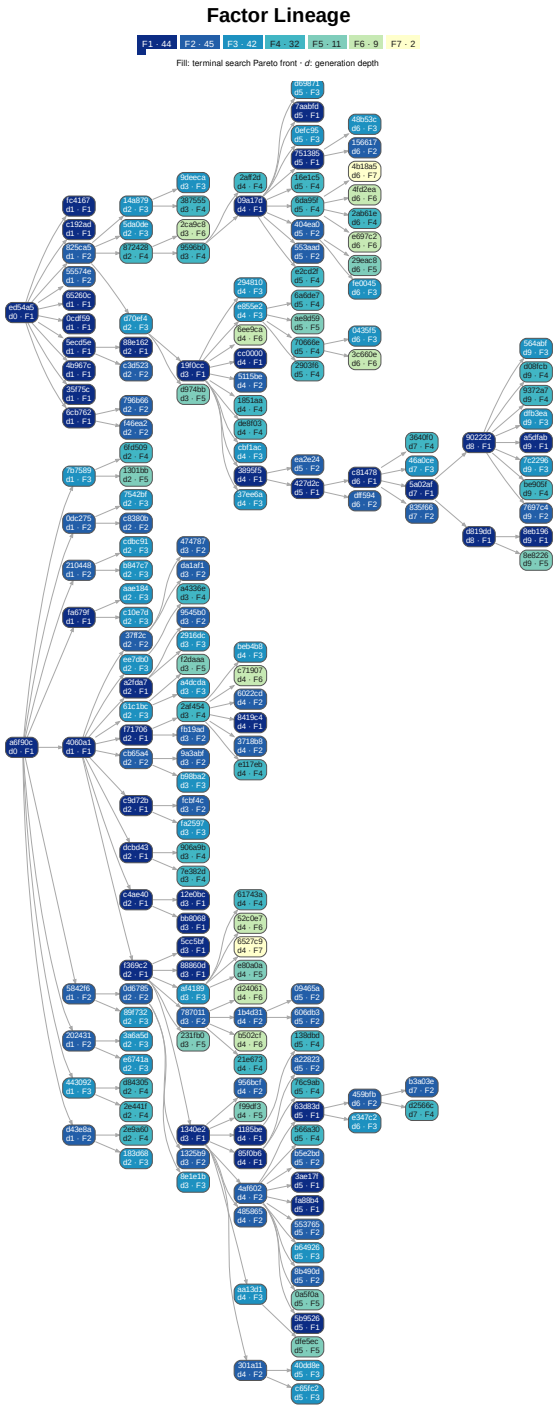


Figure 3: Factor lineage and terminal search Pareto-front assignments for the finalized CSI 300 Graph run. Node fill color marks the front; the legend reports the number of nodes in each front. Edges show direct generation links, and each node lists its shortened ID, generation depth d, and front index.

where $F_{m,i}$ is node i's propagated value for utility m. The intervals were estimated from multiple Graph runs and fixed before the ablation. Eligible parents are then ranked by

$$s_i^{\text{flow}} = \frac{z_{Q,i} + z_{F,i} + z_{D,i}}{3}. \tag{35}$$

The fixed intervals prevent the different numeric scales of the three utilities from determining the equal-weight sum. Eligible parents are ranked by descending s_i^{flow}; eligibility, search budget, and the number of selected parents follow the full model.

Flow, utility, and relation controls. Setting $\gamma = 0$ removes neighbor propagation while retaining node utilities. Utility ablations restrict search Pareto selection to (Q, D), (Q, F), or (Q) for the Fertility, Diversity, and Quality-only rows, respectively. Removing Structure or Correlation excludes that relation from both Graph Flow and AverageDistance; removing Generation excludes it from Graph Flow. The remaining Flow relation weights are renormalized.

I LLM SENSITIVITY STUDY

This appendix compares DeepSeek-V4-Flash, GPT-5.6-Sol, and Kimi K3 under the Graph, MCTS, and Chain modes of CosmosAlpha on CSI 300. All entries use the same factor-evaluation and downstream backtest conventions as the main tables.

Table 19: LLM and search-mode sensitivity on CSI 300 under the main factor-pool integration protocol. ARR and signed MDD are percentages; higher values are better for every displayed metric because a signed MDD closer to zero denotes a smaller drawdown.

Mode	LLM	Predictive Performance				Portfolio Performance		
		IC ↑	ICIR ↑	RankIC ↑	RankICIR ↑	ARR (%) ↑	IR ↑	MDD (%) ↑
Chain	DeepSeek-V4-Flash	0.0241	0.2731	0.0373	0.4260	3.94	0.2223	-35.33
	GPT-5.6-Sol	0.0280	0.2892	0.0411	0.4519	4.16	0.2599	-29.11
	Kimi K3	0.0299	0.3156	0.0401	0.3932	4.28	0.2810	-33.60
MCTS	DeepSeek-V4-Flash	0.0281	0.3238	0.0409	0.5145	5.68	0.3296	-31.56
	GPT-5.6-Sol	0.0259	0.2868	0.0419	0.4590	5.95	0.3361	-33.81
	Kimi K3	0.0289	0.3025	0.0412	0.5143	4.23	0.2502	-34.99
Graph	DeepSeek-V4-Flash	0.0324	0.3441	0.0494	0.5670	8.44	0.5332	-25.07
	GPT-5.6-Sol	0.0294	0.3379	0.0431	0.5119	7.51	0.4358	-26.59
	Kimi K3	0.0324	0.3818	0.0463	0.5909	6.11	0.3519	-28.91

Across all three backbone models, Graph delivers the strongest overall performance and leads all seven metrics for each backbone. The portfolio results particularly favor DeepSeek-V4-Flash in the Graph setting: it leads ARR, IR, and signed MDD for Graph. For MCTS, GPT-5.6-Sol has slightly higher ARR and IR than DeepSeek-V4-Flash (5.95 versus 5.68 and 0.3361 versus 0.3296), whereas DeepSeek-V4-Flash has the better signed MDD (-31.56% versus -33.81%). Across each search mode, the three backbone models remain relatively close on Predictive Performance, whereas the differences between Graph, MCTS, and Chain are more pronounced, especially for Portfolio Performance.

J INDEPENDENT-RUN STABILITY STUDY

This appendix evaluates the stability of CosmosAlpha-Graph, CosmosAlpha-MCTS, and CosmosAlpha-Chain with DeepSeek-V4-Flash on CSI 300 across three independent LLM runs. All three methods are reported in one table under the same factor-pool integration protocol as the main results. The Mean and Std. rows are computed across the three runs for each method according to Appendix B.4. The data split and experimental configuration remain fixed across repetitions, so the variation reflects stochasticity in the factors generated by the LLM.

Table 20: Three-run stability of CosmosAlpha-Graph, CosmosAlpha-MCTS, and CosmosAlpha-Chain with DeepSeek-V4-Flash on CSI 300 under the main integration protocol. The table uses the same predictive and portfolio metric groups and metric directions as Tables 1 and 2. ARR and signed MDD are percentages. The Mean and Std. rows are computed across the three independent runs for each method; Std. uses the population standard deviation.

Mode	Run	Predictive Performance				Portfolio Performance		
		IC ↑	ICIR ↑	RankIC ↑	RankICIR ↑	ARR (%) ↑	IR ↑	MDD (%) ↑
Chain	Run 1	0.0230	0.2606	0.0369	0.4087	3.49	0.1839	-39.27
	Run 2	0.0234	0.2732	0.0371	0.4261	4.06	0.2399	-33.13
	Run 3	0.0242	0.2912	0.0352	0.4332	4.48	0.2631	-36.60
	Mean	0.0235	0.2750	0.0364	0.4227	4.01	0.2290	-36.33
	Std.	0.0005	0.0126	0.0009	0.0103	0.41	0.0332	2.51
MCTS	Run 1	0.0284	0.2993	0.0452	0.5024	5.93	0.3546	-26.50
	Run 2	0.0277	0.3618	0.0398	0.5337	5.71	0.3253	-31.79
	Run 3	0.0294	0.3204	0.0384	0.5173	5.50	0.3288	-35.40
	Mean	0.0285	0.3272	0.0411	0.5178	5.71	0.3362	-31.23
	Std.	0.0007	0.0260	0.0029	0.0128	0.18	0.0131	3.65
Graph	Run 1	0.0312	0.3318	0.0485	0.5449	8.87	0.5463	-24.68
	Run 2	0.0331	0.3370	0.0512	0.5714	8.57	0.5320	-24.28
	Run 3	0.0336	0.3712	0.0506	0.5935	7.92	0.5412	-26.33
	Mean	0.0326	0.3467	0.0501	0.5699	8.45	0.5398	-25.10
	Std.	0.0010	0.0175	0.0012	0.0199	0.40	0.0059	0.89

Using the three independent runs, CosmosAlpha-Graph has the highest mean in all seven metrics. Its mean IC, ICIR, RankIC, and RankICIR are 0.0326, 0.3467, 0.0501, and 0.5699, respectively, compared with 0.0285, 0.3272, 0.0411, and 0.5178 for MCTS and 0.0235, 0.2750, 0.0364, and 0.4227 for Chain. The same ordering holds for the portfolio metrics: Graph reaches a mean ARR of 8.45%, IR of 0.5398, and signed MDD of -25.10%, outperforming MCTS and Chain on the mean risk–return profile. Run-to-run dispersion is metric-dependent. Graph's ARR standard deviation is 0.40, between MCTS (0.18) and Chain (0.41), while its IR and MDD standard deviations (0.0059 and 0.89) are lower than those of MCTS (0.0131 and 3.65) and Chain (0.0332 and 2.51). Although Graph is not the least variable method on every predictive metric, it leads every metric on average, indicating that the overall advantage is not caused by a single favorable run.