ALPHADREAMER: MODEL-PREDICTIVE PREFIX PLANNING FOR FORMULAIC ALPHA DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Discovering predictive alpha factors is fundamental to quantitative trading. However, evaluating a formula's predictive quality requires completing its construction, leaving intermediate decisions without direct performance feedback. We introduce AlphaDreamer, a model-based reinforcement learning framework that learns predictive models from evaluated formulas to guide construction through prefix-level planning. Its Model-Predictive Prefix Planner (MPPP) couples exact grammar transitions with learned predictions of prefix representations and completion quality. A learned reference model directly scores candidate prefixes to complement predictive lookahead in action selection. During prefix planning, the planner executes only the first action of each selected plan before replanning from the updated prefix. A shared policy learns both to generate formulas from scratch and to complete planner-provided prefixes. Empirical results show that AlphaDreamer outperforms the evaluated baselines in predictive accuracy and portfolio performance. Prefix-level planning improves both discovery quality and evaluation efficiency, while the full planner consistently outperforms direct prefix scoring alone.

1 INTRODUCTION

Formulaic alphas express predictive relationships in market data as mathematical programs. Their explicit structure supports interpretation and combination into factor libraries, but discovery requires searching a combinatorial space of variables, operators, and time windows. Reinforcement learning has made this search increasingly effective by learning sequential generation policies from the performance of completed formulas (Yu et al., 2023; Zhu & Zhu, 2025; Chen et al., 2026).

The central difficulty is the mismatch between when a construction decision is made and when its quality can be evaluated (Figure 1). An early operator determines the arguments that follow and constrains the remaining expression, whereas predictive performance becomes available only after the formula is complete. A learned policy incorporates this delayed feedback into its action probabilities. Predictive planning provides a complementary mechanism: it uses evaluated formulas to assess alternative continuations at the current prefix before selecting the next symbol.

Search-based methods such as Alpha^2 and AlphaCFG already combine policy and value predictions with Monte Carlo tree search (Xu et al., 2024; Yang et al., 2026). AlphaDreamer develops a model-based approach to this decision problem. Formula construction has an exact symbolic transition rule: the grammar specifies which actions are legal and how they change the expression. What remains to be learned is how these changes affect the prospects of completing a predictive formula. We therefore separate grammatical construction from learned evaluation, using a relational representation of the prefix to predict the consequences of candidate actions.

Our Model-Predictive Prefix Planner (MPPP) implements this separation through finite-horizon search. Exact prefix transitions preserve grammatical constraints, while a learned latent transition model supplies the representations used to score candidate continuations. A separate reference model directly encodes the current prefix and each candidate's symbolic successor, allowing its score to be compared with latent lookahead. Replanning after each executed action limits commitment to a single construction decision and refreshes the search from the resulting expression. This design connects terminal evaluation with decisions made before the formula is complete.

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.

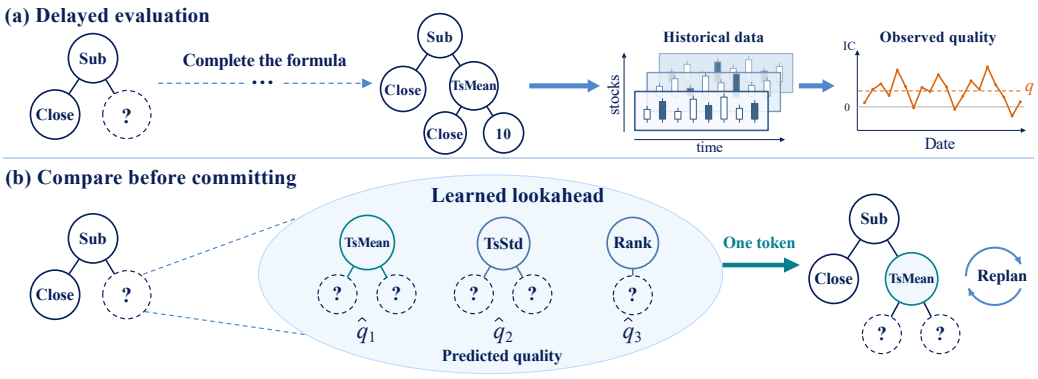

Figure 1: **Motivation for predictive prefix planning.** (a) Empirical quality is obtained after a formula is completed and evaluated on historical data. (b) Learned lookahead compares candidate continuations before committing one token and replanning. Predictions are learned from evaluated construction trajectories. Dashed nodes denote unfilled slots; q denotes observed quality and $\hat{q}_i$ predicted quality. The IC curve is schematic.

The shared policy generates formulas directly from an empty prefix or completes prefixes produced by planning. During training, evaluation of the resulting formulas supplies supervision for the predictive models and rewards for policy learning.

Our contributions are threefold:

- We introduce AlphaDreamer, a model-based reinforcement learning framework for formulaic alpha discovery. It turns evaluations of completed formulas into predictive guidance for intermediate construction decisions.
- We develop MPPP, which combines exact grammar transitions, learned predictive lookahead, and direct prefix evaluation to guide formula construction. The planner executes one action before replanning and works with a shared policy that learns both formula generation and prefix completion.
- We demonstrate that AlphaDreamer outperforms the evaluated baselines in predictive accuracy and portfolio performance. Planning improves discovery quality at matched evaluation budgets, and the full planner consistently outperforms direct prefix scoring alone.

2 Related Work

Formulaic alpha discovery. AutoAlpha explores formulaic factors through hierarchical evolutionary search (Zhang et al., 2020). AlphaGen learns to generate factors that improve a combined signal (Yu et al., 2023). AlphaQCM uses distributional reinforcement learning for exploration (Zhu & Zhu, 2025). AlphaSAGE combines relational graph representations with GFlowNet sampling (Chen et al., 2026). AlphaForge couples generation with learned fitness prediction and dynamic combination (Shi et al., 2025), while AlphaAgentEvo iteratively revises formulas through agentic generation and evaluation (Tang et al., 2026). AlphaDreamer uses completed-formula feedback to compare continuations during expression construction.

Search over symbolic expressions. Deep Symbolic Regression learns expression generation through risk-seeking policy gradients (Petersen et al., 2021). In alpha discovery, Alpha^2 and AlphaCFG use neural Monte Carlo tree search (Xu et al., 2024; Yang et al., 2026), while RiskMiner incorporates risk-seeking search (Ren et al., 2024). AlphaDreamer instead uses a learned transition model to predict the next prefix embedding for each candidate action. These predictions guide beam search, while exact grammar transitions keep the corresponding expressions valid.

Model-based reinforcement learning. PETS learns dynamics for model-predictive control (Chua et al., 2018), MuZero combines latent dynamics with search (Schrittwieser et al., 2020), and Dreamer

learns behavior through imagined trajectories (Hafner et al., 2025). TD-MPC combines short-horizon latent planning with a terminal value estimate learned through temporal differences (Hansen et al., 2022). AlphaDreamer's symbolic transitions are known; its predictive heads learn completed-formula quality from evaluated trajectories. Model rollouts then guide symbolic construction actions.

3 METHOD

AlphaDreamer turns the evaluation of completed formulas into guidance for their construction. As shown in Figure 2, evaluated trajectories train models of prefix representations and completion quality. MPPP uses these models to select construction actions. A shared policy completes the resulting prefixes and supplies further evaluated formulas. The method separates three responsibilities: the grammar preserves symbolic validity, the predictive models assess candidate continuations, and the policy learns generation and completion behavior from market feedback.

3.1 PROBLEM FORMULATION

Formula quality. Let $\mathcal{D} = \{(X_d, \mathcal{U}_d, y_d)\}_{d=1}^{T}$ be a historical market sample, where X_d contains the information available at date d, $\mathcal{U}_d$ is the stock universe, and $y_d(i)$ is the subsequent return of stock i. A formula f produces a cross-sectional signal $f_d(i)$. We measure its signed predictive quality on a set of dates D by

$$q_D(f) = \frac{1}{|D|} \sum_{d \in D} \mathrm{Corr}_{i \in \mathcal{U}_d}(f_d(i), y_d(i)) \,. \tag{1}$$

Here Corr is Pearson correlation over finite signal–target pairs, and the prediction target is the twenty-trading-day forward return. This quality measure provides the learning signal for the predictive models and prefix ranker that guide formula construction.

Formula construction. At construction step t, the state s_t is the current token prefix, with $s_0 = \varnothing$. The vocabulary contains features, constants, operators, time windows, and `STOP`. A typed grammar $\mathcal{G}$ defines the admissible tokens and their deterministic symbolic transition:

$$a_t \in \mathcal{A}_{\mathcal{G}}(s_t), \qquad s_{t+1} = T_{\mathcal{G}}(s_t, a_t). \tag{2}$$

Each nonterminal action fills a pending argument subject to type, depth, and length constraints. `STOP` ends construction after a complete expression has been formed. A completed trajectory τ yields a formula f_τ, whose training-period quality provides supervision for the prefixes and actions along that trajectory. The policy $\pi_\theta(a \mid s)$ is masked to assign zero probability to grammar-invalid actions.

3.2 LEARNING PREFIX CONSEQUENCES

Relational representation. Subtraction distinguishes operand order, while a time-series operator distinguishes its series input from its window length. Following the relational expression encoding of AlphaSAGE (Chen et al., 2026), we parse each prefix into $G(s) = (V_s, E_s)$ and apply an RGCN (Schlichtkrull et al., 2018). The relation set $\mathcal{R}$ contains unary, commutative, left, right, series, and window edges and their reverses (Tables 4–5). At layer ℓ, node v is updated by

$$h_v^{(\ell+1)} = \mathrm{ReLU}\left(\mathrm{LN}\left[W_0^{(\ell)} h_v^{(\ell)} + b_0^{(\ell)} + \sum_{r \in \mathcal{R}} \sum_{u \in \mathcal{N}_r(v)} W_r^{(\ell)} h_u^{(\ell)} \right] \right), \tag{3}$$

where $\mathcal{N}_r(v)$ denotes incoming neighbors under relation r, $W_r^{(\ell)}$ transforms their messages, and $W_0^{(\ell)}$ and $b_0^{(\ell)}$ define the self-node transformation and bias. Initial states $h_v^{(0)}$ are token embeddings. LN denotes layer normalization. Mean pooling after L layers gives the d_z-dimensional representation

$$z_\phi(s) = E_\phi(G(s)) = \frac{1}{|V_s|} \sum_{v \in V_s} h_v^{(L)}. \tag{4}$$

Only generated tokens enter the graph. An empty prefix uses a padding node. The actor–critic and predictive models use separately parameterized encoders.

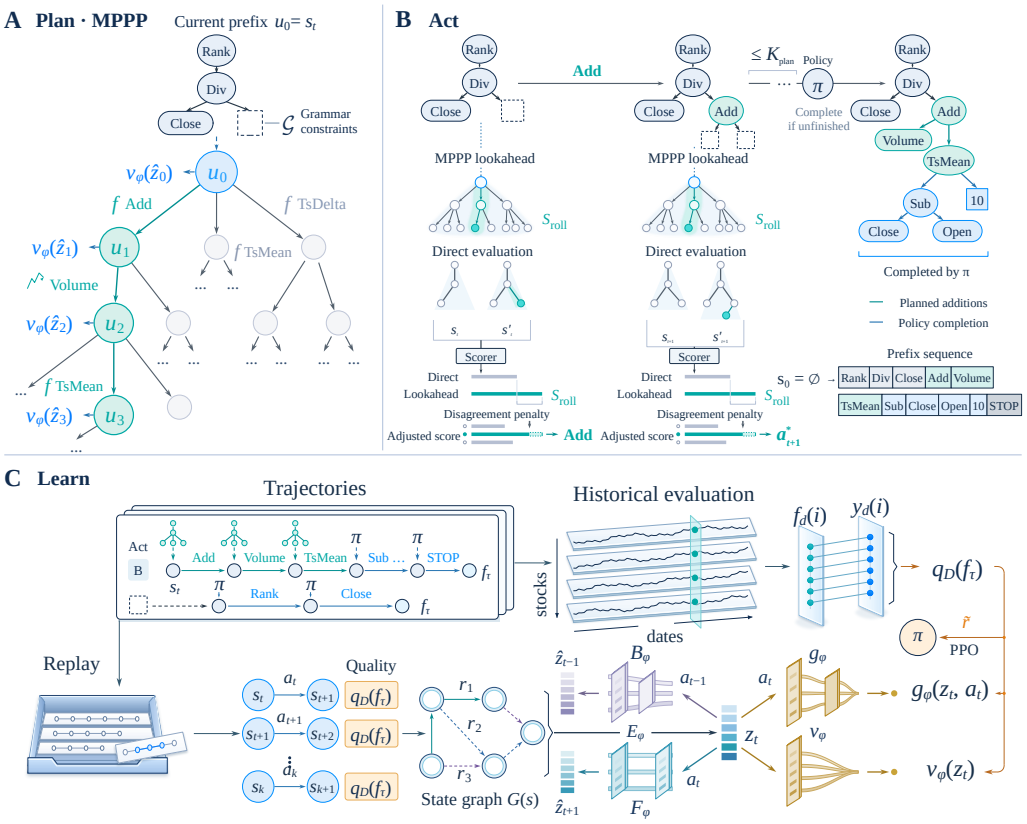

Figure 2: **AlphaDreamer framework. (A) Plan:** Model-predictive prefix planning (MPPP) explores grammar-valid continuations using predicted prefix representations and completion-quality estimates. **(B) Act:** Disagreement between lookahead and direct evaluation penalizes action scores. MPPP executes the selected token and replans for up to K_{plan} steps; the shared policy π then completes any unfinished formula. **(C) Learn:** Historical evaluation assigns a signed quality $q_D(f_\tau)$ to each completed formula, shared by transitions along its trajectory. Replay trains the encoder E_φ, backward and forward predictors (B_φ, F_φ), and action-conditioned and state-conditioned quality heads (g_φ, v_φ). Shaped rewards train π with PPO using only policy-sampled steps.

Transition and quality learning. Evaluated trajectories supply tuples $(s_t, a_t, s_{t+1}, q_D(f_\tau))$. For $z = z_\phi(s_t)$ and $z^+ = z_\phi(s_{t+1})$, planning uses predictions of both the successor representation and completion quality. The forward model $F_\phi(z, a_t)$ predicts z^+ without re-encoding each candidate graph. The heads $g_\phi(z, a)$ and $v_\phi(z)$ predict signed completion quality $q_D(f_\tau)$ conditional on a proposed action and on the prefix, respectively: g_ϕ scores extensions, and v_ϕ evaluates the search boundary.

The backward model $B_\phi(z^+, a_t)$ reconstructs z to constrain adjacent-prefix representations during training. An auxiliary head $p_\phi(z, a) \in (0, 1)$, trained on accepted completions, supplies a bounded term for scoring and filtering. The grammar determines action legality. The joint training objective and coefficients are given in Appendix A.2.

Starting prefixes. A ranker $\rho_\psi(s)$ with a separate graph encoder and scalar readout prioritizes reusable prefixes extracted from evaluated formulas and trajectories. Each prefix inherits its source completion's signed quality. For prefixes s^+, s^- with source qualities $q^+ > q^-$, the pairwise logistic loss (Burges et al., 2005) is

$$\mathcal{L}_{\text{seed}} = -\log \text{sigmoid}\big(\rho_\psi(s^+) - \rho_\psi(s^-)\big)\,. \tag{5}$$

This objective concentrates planning on promising starting structures. The planner's predictive and reference scores then guide their extension. The ranker does not enter continuation scoring (Appendix A.4).

3.3 Model-Predictive Prefix Planning

MPPP uses learned completion-quality predictions to compare the consequences of a token choice before committing to it. The search horizon H controls the depth of each comparison. A separate budget K_{plan} controls the number of actions executed by the planner before the shared policy completes the expression.

Grammar-constrained lookahead. At the current prefix s_t, a search begins with $u_0 = s_t$ and $\widehat{z}_0 = z_\phi(s_t)$. A candidate action advances its symbolic prefix and predicted representation together:

$$\begin{aligned} a_h &\in \mathcal{A}_{\mathcal{G}}(u_h), \\ u_{h+1} = T_{\mathcal{G}}(u_h, a_h), &\qquad \widehat{z}_{h+1} = F_\phi(\widehat{z}_h, a_h). \end{aligned} \tag{6}$$

The symbolic prefix supplies legal actions and terminal conditions at every depth. The predicted representation supplies the input for subsequent quality estimates. This division lets search use learned consequences while maintaining an explicit, grammar-valid expression along each branch.

Continuation scoring. For a candidate $\xi = (a_0, \ldots, a_{k-1})$ with $k \le H$, we use

$$J_\phi(\xi \mid s_t) = \sum_{h=0}^{k-1} \gamma^h \left[g_\phi(\widehat{z}_h, a_h) + \beta p_\phi(\widehat{z}_h, a_h) \right] + \gamma^k v_\phi(\widehat{z}_k). \tag{7}$$

As a formula takes shape, successive prefixes reveal the consequences of construction choices. Action-conditioned predictions assess completion quality at each imagined step, while the boundary estimate summarizes completion prospects after the candidate sequence. We aggregate these assessments into the path utility J_ϕ, favoring paths that remain promising throughout lookahead and using their collective support to select the next token. The depth weight $\gamma \in (0, 1]$ controls later predictions, and $\beta \ge 0$ weights the auxiliary term.

Beam search retains the M entries with highest J_ϕ at each depth. Terminal entries are scored but not expanded. Let $\mathcal{C}_t$ contain all candidates generated across depths $1, \ldots, H$, recorded before pruning. For cross-depth comparison, we use the length-normalized score $\bar{J}_\phi$ defined in Appendix B, preserving the original one-step scale. The rollout score is

$$S_{\text{roll}}(s_t, a) = \max_{\xi \in \mathcal{C}_t :\, \xi_0 = a} \bar{J}_\phi(\xi \mid s_t). \tag{8}$$

This maximization evaluates a token by the best continuation discovered from it, allowing an action to remain competitive even when other continuations beginning with the same token are poor.

Direct prefix evaluation. Latent lookahead evaluates states produced by the learned transition model. We also assess each candidate first action through a separate encoder applied directly to the current and successor expressions. This provides a reference for the immediate decision that does not depend on recursively predicted representations. Define

$$z^\omega = E_\omega(G(s_t)), \qquad z^{\omega,a} = E_\omega(G(T_{\mathcal{G}}(s_t, a))).$$

The reference score is

$$S_{\text{ref}}(s_t, a) = g_\omega(z^\omega, a) + \gamma v_\omega(z^{\omega,a}) + \beta_{\text{ref}} p_\omega(z^\omega, a). \tag{9}$$

The reference heads are trained on the same completed-formula labels as the predictive model, with separate parameters. Their direct access to the candidate expression supplies an additional assessment of the action being considered for execution. The training objective and β_{ref} are given in Appendix A.3.

Selection and replanning. We combine rollout support with the reference assessment using a score-discrepancy penalty:

$$\begin{aligned} \Delta(s_t, a) &= \left| S_{\text{roll}}(s_t, a) - S_{\text{ref}}(s_t, a) \right|, \\ \widetilde{S}(s_t, a) &= S_{\text{roll}}(s_t, a) - \lambda \Delta(s_t, a), \\ a_t^* &= \arg \max_{a \in \mathcal{A}_{\text{adm}}(s_t)} \widetilde{S}(s_t, a). \end{aligned} \tag{10}$$

Algorithm 1 Model-Predictive Prefix Planning

Input: Prefix s, planning horizon H, beam width M, execution budget K_{plan}, and learned parameters ϕ and ω.
Output: A completed expression, or a prefix to be completed by π_θ.

for $k = 1, \dots, K_{\text{plan}}$ **do**
  $u_0 \leftarrow s$, $\widehat{z}_0 \leftarrow z_\phi(s)$, $\mathcal{C} \leftarrow \varnothing$.
  Initialize the beam with the empty continuation rooted at $(u_0, \widehat{z}_0)$.
  for $h = 0, \dots, H-1$ **do**
    $\mathcal{C}_{\text{new}} \leftarrow \varnothing$.
    for each nonterminal beam entry $(\xi, u_h, \widehat{z}_h)$ and each $a_h \in \mathcal{A}_{\mathcal{G}}(u_h)$ **do**
      Append a_h to ξ to form ξ'.
      $u_{h+1} \leftarrow T_{\mathcal{G}}(u_h, a_h)$, $\widehat{z}_{h+1} \leftarrow F_\phi(\widehat{z}_h, a_h)$.
      Compute $J_\phi(\xi' \mid s)$ and $\bar{J}_\phi(\xi' \mid s)$ using Eqs. (7) and (B.1).
      Add $(\xi', u_{h+1}, \widehat{z}_{h+1})$, together with both scores, to $\mathcal{C}_{\text{new}}$.
    end for
    if $\mathcal{C}_{\text{new}} = \varnothing$ **then break**.
    Record all candidates before pruning: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\text{new}}$.
    Replace the beam with up to M entries of $\mathcal{C}_{\text{new}}$ having the highest J_ϕ.
    Mark `STOP` entries terminal.
  end for
  Compute $S_{\text{ref}}(s, a)$ for all $a \in \mathcal{A}_{\mathcal{G}}(s)$ using Eq. (9).
  for each first action a represented in $\mathcal{C}$ **do**
    $S_{\text{roll}}(s, a) \leftarrow \max_{\xi \in \mathcal{C}:\, \xi_0 = a} \bar{J}_\phi(\xi \mid s)$.
    $\Delta(s, a) \leftarrow |S_{\text{roll}}(s, a) - S_{\text{ref}}(s, a)|$.
    $\widetilde{S}(s, a) \leftarrow S_{\text{roll}}(s, a) - \lambda \Delta(s, a)$.
  end for
  Form the admissible action set $\mathcal{A}_{\text{adm}}(s)$ using Eq. (B.3).
  if $\mathcal{A}_{\text{adm}}(s) \neq \varnothing$ **then**
    $a^* \leftarrow \arg\max_{a \in \mathcal{A}_{\text{adm}}(s)} \widetilde{S}(s, a)$.
  else
    $a^* \leftarrow \arg\max_{a \in \mathcal{A}_{\mathcal{G}}(s)} S_{\text{ref}}(s, a)$.
  end if
  if $a^* =$ `STOP` **then return** the completed expression represented by s.
  Execute a^*: $s \leftarrow T_{\mathcal{G}}(s, a^*)$.
end for
return s for completion by π_θ.

The penalty ties the preference induced by lookahead to a direct assessment of the next expression, reducing the priority of actions for which the two evaluators disagree. Its strength is controlled by $\lambda \geq 0$. The admissible set $\mathcal{A}_{\text{adm}}(s_t)$ applies the auxiliary-score and discrepancy thresholds specified in Appendix B. If this set is empty, the planner selects the legal action maximizing S_{ref}.

Only a_t^* is executed. The resulting prefix is re-encoded before the next search, so the remaining construction is reconsidered from the updated symbolic state. This limits commitment to the next token and replaces the imagined root with a directly encoded prefix at every decision. Planning ends when `STOP` is selected or the K_{plan} budget is exhausted; the shared policy completes any unfinished expression. Algorithm 1 summarizes this procedure.

3.4 Policy Learning from Planned Prefixes

Planning and policy learning serve complementary roles. MPPP directs construction toward prefixes favored by predictive evaluation, while the policy learns how to turn those prefixes into complete formulas. Training on both empty and planner-provided prefixes gives the same policy access to generation and conditional completion, with market evaluation providing feedback in both cases.

The terminal reward uses signed training-period quality $q_D(f)$ together with novelty and redundancy terms relative to an evolving factor library $\mathcal{L}$. Quality rewards predictive formulas; novelty favors distinct signals; and correlation, duplicate, and family penalties control repeated discovery. The complete shaping function is given in Appendix A.5.

We optimize the shared actor–critic with clipped PPO (Schulman et al., 2017) on fresh policy-sampled steps from direct generation and completion of planned prefixes. Planner-selected transitions

contribute to predictive-model training but are excluded from PPO, as are imagined rollouts. Each collection batch is followed by a policy update before the next batch is sampled.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Evaluation metrics. We evaluate predictive performance using the Information Coefficient (IC), Rank Information Coefficient (RIC), and their information ratios (ICIR and RICIR). AlphaDreamer fixes each factor's direction using its mean validation IC before each test year. Portfolio performance is measured by annualized return (AR) and the Sharpe ratio (SR). We report mean $\pm$ sample standard deviation across three random seeds.

Datasets and evaluation. Experiments cover CSI300 and CSI500 in the Chinese market and S&P 500 in the U.S. market, with the 20-trading-day forward return as the prediction target. All methods use the same chronological training, validation, and test splits. For the 2023 evaluation, training covers 2010–2020 and validation covers 2021–2022. For each subsequent test year through 2025, the training period expands by one year and the two-year validation window moves forward by one year. Each method is retrained and validated before evaluation on the following year, using only information available at that time. This design better reflects quantitative investment practice, where models are periodically updated to incorporate new information and changing market conditions, than keeping a single model fixed throughout a multi-year test period.

Baselines. We compare AlphaDreamer with predictive models (MLP, LightGBM (Ke et al., 2017), and XGBoost (Chen & Guestrin, 2016)), genetic programming (GP), and formulaic alpha discovery methods: AlphaGen (Yu et al., 2023), AlphaQCM (Zhu & Zhu, 2025), AlphaSAGE (Chen et al., 2026), and AlphaForge (Shi et al., 2025). Data, metric, and training details appear in Appendix C.

4.2 MAIN RESULTS: DOES ALPHADREAMER DISCOVER BETTER ALPHAS?

Table 1 shows that AlphaDreamer achieves the highest mean IC, ICIR, RIC, RICIR, annualized return, and Sharpe ratio across all three markets. Its IC reaches 9.53% on CSI300, 4.69% on CSI500, and 5.87% on S&P 500, exceeding the strongest baseline in each market by 83.6%, 27.4%, and 36.2%, respectively.

Figure 3 compares cumulative net returns across the three markets. AlphaDreamer achieves the highest cumulative net return in each market, complementing its lead in the correlation metrics.

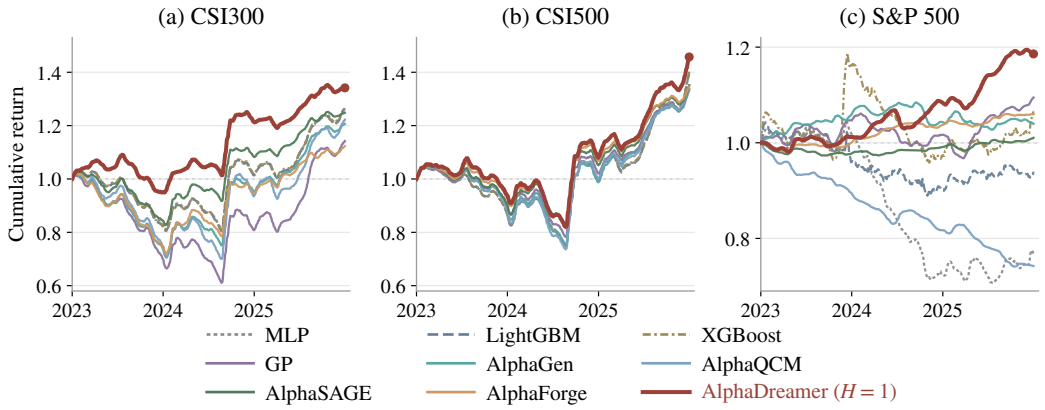

Figure 3: Cumulative net returns across three markets (2023–2025), averaged over three seeds. Return construction is detailed in Appendix C.3.

Table 1: Out-of-sample performance (2023–2025; mean ± sample standard deviation over three seeds). Bold marks the highest mean in each market; AlphaDreamer uses $H = 1$.

	Correlation metrics				Portfolio metrics	
Method	IC (%) ↑	ICIR ↑	RIC (%) ↑	RICIR ↑	AR (%) ↑	SR ↑
(a) CSI300						
MLP	2.52±0.97	0.1846±0.0745	2.54±0.91	0.2020±0.0800	9.34±0.20	2.124±0.038
LightGBM	1.77±0.06	0.1375±0.0066	3.07±0.07	0.2637±0.0088	9.18±0.04	2.062±0.005
XGBoost	1.41±0.01	0.1056±0.0016	2.60±0.09	0.2236±0.0073	9.28±0.04	2.075±0.007
GP	5.19±3.40	0.3653±0.2180	5.21±3.66	0.3113±0.1958	4.92±1.20	0.91±0.16
AlphaGen	2.34±0.37	0.1678±0.0283	3.02±0.55	0.1963±0.0296	7.16±1.27	1.53±0.32
AlphaQCM	3.18±1.21	0.2404±0.0706	3.40±0.64	0.2267±0.0343	7.70±3.14	1.37±0.56
AlphaSAGE	3.53±1.84	0.2327±0.0992	4.51±2.14	0.2941±0.1265	8.61±3.32	2.28±1.05
AlphaForge	2.92±0.15	0.1763±0.0134	2.18±1.07	0.1186±0.0608	4.27±1.04	1.01±0.22
AlphaDreamer	**9.53**±2.17	**0.5349**±0.0397	**9.51**±1.16	**0.5204**±0.0322	**11.84**±2.80	**3.16**±0.53
(b) CSI500						
MLP	2.11±0.18	0.1999±0.0137	2.29±0.80	0.1989±0.0799	13.95±0.29	2.672±0.046
LightGBM	2.13±0.09	0.2191±0.0092	3.65±0.10	0.2899±0.0082	13.70±0.01	2.611±0.002
XGBoost	1.76±0.10	0.1661±0.0113	2.84±0.05	0.2298±0.0065	13.84±0.04	2.632±0.006
GP	1.38±0.33	0.1040±0.0309	3.93±1.11	0.3235±0.1170	11.55±2.41	2.34±0.40
AlphaGen	1.94±1.18	0.1506±0.0885	3.40±1.09	0.2275±0.0744	11.68±0.88	2.33±0.21
AlphaQCM	1.52±0.36	0.1347±0.0104	2.99±1.57	0.1842±0.0916	12.17±1.55	2.31±0.45
AlphaSAGE	3.10±0.70	0.2080±0.0321	5.26±0.94	0.3352±0.0496	12.17±0.98	2.61±0.28
AlphaForge	3.68±0.30	0.2025±0.0158	5.17±0.57	0.2719±0.0389	12.06±0.93	2.60±0.18
AlphaDreamer	**4.69**±0.87	**0.3101**±0.0928	**5.63**±0.48	**0.3426**±0.0223	**15.85**±4.78	**3.01**±0.61
(c) S&P 500						
MLP	3.45±0.36	0.2743±0.0641	1.97±0.14	0.1630±0.0281	−7.50±2.68	−2.410±0.759
LightGBM	2.48±0.27	0.1893±0.0141	1.15±0.03	0.0920±0.0022	−2.03±2.84	−0.465±0.659
XGBoost	2.59±0.13	0.1962±0.0126	1.56±0.01	0.1192±0.0006	2.13±1.00	0.412±0.203
GP	3.58±1.24	0.2544±0.0866	2.15±0.50	0.1631±0.0442	3.17±2.25	1.91±1.21
AlphaGen	3.62±2.32	0.3264±0.1861	2.35±0.70	0.2029±0.0426	1.33±1.15	1.05±0.93
AlphaQCM	4.31±0.46	0.3762±0.0609	2.92±0.33	0.2722±0.0508	−8.64±2.62	−7.10±2.98
AlphaSAGE	1.98±0.96	0.2610±0.0715	1.54±0.28	0.2454±0.0338	0.35±2.43	0.96±3.62
AlphaForge	2.36±1.00	0.2378±0.1251	0.83±0.35	0.1402±0.0493	2.03±2.41	3.67±3.72
AlphaDreamer	**5.87**±1.18	**0.5215**±0.0991	**3.04**±0.03	**0.2989**±0.0128	**6.23**±2.50	**4.90**±2.27

4.3 Model-Predictive Planning Drives Discovery Quality

One-step planning improves discovery quality and evaluator efficiency.

Table 2: One-step planning on CSI300.

Metric	Policy-only	$H = 1$
IC (%) ↑	8.69±0.60	**9.53**±2.17
RIC (%) ↑	7.66±0.57	**9.51**±1.16
$\text{IC}_{\max}$@80% (%) ↑	11.33±0.61	**11.53**±0.23
$\text{RIC}_{\max}$@80% (%) ↑	11.01±0.52	**11.63**±0.93
Evaluator calls ↓	8,221±277	**6,141**±133
Unique valid formulas ↑	1,035±382	**1,558**±256
Unique valid / 1k calls ↑	125±43	**252**±47

Table 3: Planning-horizon ablation on CSI300.

H	IC (%) ↑	RIC (%) ↑	Unique valid / 1k calls ↑	Planning time (ms) ↓
1	**9.53**±2.17	**9.51**±1.16	**252.3**±47.2	**20.53**±0.16
2	9.49±1.51	8.53±2.53	213.3±39.2	41.21±2.99
3	7.39±0.84	7.00±1.42	201.5±17.1	66.09±4.36
4	8.50±2.05	8.25±1.34	229.2±44.3	91.65±10.65
5	8.29±0.57	7.01±0.93	216.7±27.5	115.94±12.10
6	9.25±0.78	8.89±1.13	221.3±43.2	143.91±13.16
7	8.03±1.98	7.15±1.65	249.4±64.6	162.85±21.32
8	9.02±1.69	8.89±2.51	215.4±31.3	194.70±22.74
9	8.95±0.67	7.88±0.22	204.0±12.7	223.60±26.86

Mean ± sample standard deviation across three seeds, after averaging years within each seed. Standard deviations are shown in a smaller font. Bold marks the best mean. Planning time is per decision.

Table 2 shows that one-step planning increases mean IC from 8.69% to 9.53% and mean RankIC from 7.66% to 9.51%, corresponding to relative gains of 9.7% and 24.2%. It also reduces actual evaluator calls by approximately 25% while producing 51% more unique valid formulas. The mean yield rises from 125 to 252 unique valid formulas per 1,000 calls. Predictive action selection therefore improves both the quality of the discovered signals and the productivity of formula evaluation.

Table 3 shows that one-step lookahead ($H = 1$) achieves the highest mean IC, RankIC, and evaluator yield among the tested horizons. Increasing the horizon from $H = 1$ to $H = 9$ raises mean decision time from 20.53 to 223.60 ms without consistent improvements in discovery quality. In these experiments, repeated one-step lookahead offers the best balance of discovery quality, evaluator yield, and planning time.

Figure 4(a,b) shows that planning surpasses policy-only generation in both IC and RankIC at every evaluator budget. At 5,000 calls, it reaches 9.60% IC and 9.56% RankIC, versus 8.09% and 7.10% for policy-only generation. The complete planner also beats direct reference scoring in every one of six paired seeds, for both IC and ICIR. Mean IC rises from 9.41% to 11.63%, a gain of 2.21 percentage points, while ICIR rises from 0.467 to 0.515. Together, these comparisons establish that planning improves the quality obtained from a fixed evaluation budget and that combining learned lookahead with the reference assessment is more effective than using the reference score alone.

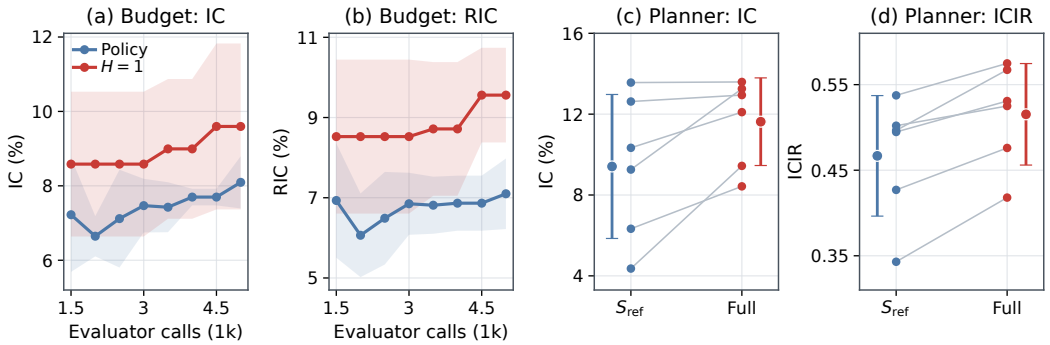

Figure 4: Planning improves discovery quality, and full MPPP improves on direct reference scoring. (a,b) Test IC and RankIC versus real evaluator calls for policy-only generation and $H = 1$ planning. Lines show the three-seed mean and shading shows sample standard deviation after averaging 2023–2025 within each seed. (c,d) Test IC and ICIR for six paired seeds comparing S_{ref}-only selection with full MPPP on the 2025 test split. Thin lines join paired results and large markers show mean $\pm$ sample standard deviation.

5 Conclusion

AlphaDreamer connects the evaluation of completed formulas with decisions made during symbolic construction. Exact grammar transitions preserve valid expressions, learned prefix models support predictive action selection, and a shared policy learns generation and completion from market feedback. Across three market benchmarks, the resulting framework achieves the strongest performance among the evaluated methods on all reported correlation and portfolio metrics. Planning ablations further show that repeated one-step lookahead improves discovery quality and evaluator yield, while greater search depth adds cost without consistent gains. Across six paired seeds, the complete planner also outperforms direct reference scoring in both IC and ICIR. Together, these results show that predictive evaluation of candidate continuations can improve grammar-constrained search for formulaic alphas.

AI Assistance Disclosure

Generative AI tools helped polish the manuscript's language and clarity and identify AlphaCFG as related work through literature retrieval. The authors reviewed the AI-assisted material and take responsibility for the paper's content, claims, citations, and results.

REFERENCES

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005. doi: 10.1145/1102351.1102363.

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 *International Conference on Learning Representations*, 2026.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/1912.04871.

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, 2024. doi: 10.1145/3677052.3698613.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pp. 593–607, 2018. URL https://arxiv.org/abs/1703.06103.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020. doi: 10.1038/s41586-020-03051-4.

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

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Seco. AlphaForge: A framework to mine and dynamically combine formulaic alpha factors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12521–12530, 2025. doi: 10.1609/aaai.v39i12.33365.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *International Conference on Learning Representations*, 2026.

Feng Xu, Yan Yin, Xinyu Zhang, Tianyuan Liu, Shengyi Jiang, and Zongzhang Zhang. Alpha^2: Discovering logical formulaic alphas using deep reinforcement learning. *arXiv preprint arXiv:2406.16505*, 2024.

Han Yang, Dong Hao, Zhuohan Wang, Qi Shi, and Xingtong Li. Alpha discovery via grammar-guided learning and search. *arXiv preprint arXiv:2601.22119*, 2026.

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, 2023. doi: 10.1145/3580305.3599831.

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.

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*, 2025.

A Training Objectives and Implementation

A.1 Operator vocabulary and graph construction

The grammar uses six market features, constants $\{\pm 30, \pm 10, \pm 5, \pm 2, \pm 1, \pm 0.5, \pm 0.01\}$, and windows $\{10, 20, 30, 40, 50\}$. Table 4 defines the AlphaSAGE-compatible operators following the evaluator's arithmetic conventions. `STOP` terminates construction.

Table 4: Features and numerical operators. F: feature; U: unary; B: binary; R: one series plus a window; P: two series plus a window. For a trailing window $x_{d-w+1:d}$, μ_w and s_w^2 denote its mean and sample variance, and $m_k = w^{-1}\sum_{j=0}^{w-1}(x_{d-j} - \mu_w)^k$. Operations are elementwise unless a cross-sectional or temporal reduction is specified.

Name	Type	Definition
Open, High, Low, Close	F	Daily opening, high, low, and closing prices.
Volume, VWAP	F	Trading volume and volume-weighted average price.
Abs	U	$\lvert x\rvert$.
SLog1p	U	$\operatorname{sign}(x)\log(1+\lvert x\rvert)$.
Inv, Sign, Log	U	$1/x$, $\operatorname{sign}(x)$, and $\log x$, respectively.
Rank	U	Cross-sectional zero-based ordinal rank divided by the number of non-missing values.
Add, Sub, Mul, Div	B	$x+y$, $x-y$, xy, and x/y, respectively.
Pow	B	x^y.
Greater, Less	B	$\max(x,y)$ and $\min(x,y)$, respectively.
Ref	R	Lagged value x_{d-w}.
TsMean, TsSum	R	μ_w and $\sum_{j=0}^{w-1} x_{d-j}$.
TsStd, TsVar	R	Sample standard deviation s_w and variance s_w^2.
TsIr	R	Window mean divided by sample standard deviation, μ_w/s_w.
TsMinMaxDiff	R	Window maximum minus window minimum.
TsMaxDiff, TsMinDiff	R	Current value minus the window maximum or minimum.
TsSkew	R	Window skewness, $m_3/m_2^{3/2}$.
TsKurt	R	Window kurtosis statistic, $m_4/(s_w^2)^2 - 3$.
TsMax, TsMin	R	Window maximum and minimum.
TsMed	R	Window median (the lower middle value when w is even).
TsMad	R	Mean absolute deviation from the window mean, $w^{-1}\sum_{j=0}^{w-1}\lvert x_{d-j} - \mu_w\rvert$.
TsRank	R	Descending rank of x_d in a finite window: $(n_> + n_\geq + 1)/(2w)$, where $n_>$ and $n_\geq$ count window values above or at least x_d.
TsDelta	R	$x_d - x_{d-w}$.
TsDiv	R	Current value divided by the window mean, x_d/μ_w.
TsPctChange	R	$(x_d - x_{d-w+1})/x_{d-w+1}$.
TsWMA	R	Normalized weighted mean, with weights $0, 1, \ldots, w-1$ from oldest to newest.
TsEMA	R	Normalized exponentially weighted mean, with weights $a^w, \ldots, a$ from oldest to newest, $a = 1 - 2/(w+1)$.
TsCov	P	Rolling sample covariance of the two input series.
TsCorr	P	Rolling Pearson correlation. If either centered sum of squares is below 10^{-6}, the denominator is set to one.

Table 5 defines six argument relations. Their reverses give $|\mathcal{R}| = 12$. Only Add/Mul share a commutative relation. Greater/Less and TsCov/TsCorr retain left/right positions despite their mathematical symmetry. The self-node transformation in Eq. (3) is separate from these edges.

Table 5: Base relations in the prefix graph. Edges point from a generated argument (or subexpression root) to its parent operator. Each relation has a separately parameterized reverse. Operator categories refer to Table 4.

Relation	Argument role	Example incoming edge
`unary`	Sole argument of a U operator.	$x \to \mathrm{Abs}$ in $\mathrm{Abs}(x)$.
`commutative`	Either argument of Add or Mul.	$x, y \to \mathrm{Add}$ in $\mathrm{Add}(x, y)$.
`left`	First argument of other B operators; first series of P operators.	$x \to \mathrm{Sub}$ in $\mathrm{Sub}(x, y)$; $x \to \mathrm{TsCorr}$ in $\mathrm{TsCorr}(x, y, w)$.
`right`	Second argument of other B operators; second series of P operators.	$y \to \mathrm{Sub}$ in $\mathrm{Sub}(x, y)$; $y \to \mathrm{TsCorr}$ in $\mathrm{TsCorr}(x, y, w)$.
`series`	Series argument of an R operator.	$x \to \mathrm{TsMean}$ in $\mathrm{TsMean}(x, w)$.
`window`	Window argument of R or P operators.	$w \to \mathrm{TsMean}$ or $w \to \mathrm{TsCorr}$.

Unfilled arguments have no placeholder nodes; an empty prefix uses one padding node. Thus `Sub TsMean Close` contains only `Close` $\xrightarrow{\text{series}}$ `TsMean` $\xrightarrow{\text{left}}$ `Sub` and reverse edges: the window of TsMean and the right argument of Sub remain unfilled.

A.2 Predictive models

The prefix encoder follows Eqs. (3)–(4) with the relations in Table 5.

For a transition from an accepted, evaluated trajectory, let $z = z_\phi(s_t)$, $z^+ = z_\phi(s_{t+1})$, and $q = q_D(f_\tau)$. The predictive-model loss is

$$\begin{aligned}\mathcal{L}_{\text{pred}} = &\frac{\|F_\phi(z, a_t) - \mathrm{sg}(z^+)\|_2^2 + \|B_\phi(z^+, a_t) - \mathrm{sg}(z)\|_2^2}{d_z} \\ &+ [g_\phi(z, a_t) - q]^2 + \lambda_{\text{cur}}[v_\phi(z) - q]^2 + \lambda_{\text{next}}[v_\phi(z^+) - q]^2 \\ &- \lambda_p \log p_\phi(z, a_t).\end{aligned} \tag{A.1}$$

Here sg stops gradients through target embeddings. We use $(\lambda_{\text{cur}}, \lambda_{\text{next}}, \lambda_p) = (0.5, 0.25, 0.2)$ and average the loss over sampled transitions. The representation terms train the forward and backward models on observed adjacent prefixes. The quality terms use the same completed-formula outcome for the action-conditioned prediction and the current and successor prefix predictions.

Model-training examples are drawn from accepted completions. Their auxiliary target is therefore one, and binary cross-entropy reduces to $-\log p_\phi(z, a_t)$. The resulting head supplies a bounded auxiliary score for the planner; the grammar determines action legality.

A.3 Direct prefix evaluation

The reference model has a separate graph encoder E_ω and heads $g_\omega, v_\omega, p_\omega$. For $z^\omega = E_\omega(G(s_t))$ and the observed completion-quality target q, its loss is

$$\mathcal{L}_{\text{ref}} = [g_\omega(z^\omega, a_t) - q]^2 + [v_\omega(z^\omega) - q]^2 - 0.2 \log p_\omega(z^\omega, a_t). \tag{A.2}$$

The score in (9) combines the action-conditioned prediction at the current prefix with a prefix prediction obtained by directly encoding the candidate successor. Its auxiliary coefficient is $\beta_{\text{ref}} = 0.05$.

A.4 Starting-prefix ranking

The ranker uses the pairwise objective in Eq. (5) and a separate graph encoder and readout. Its score selects starting structures but is not added to the continuation score in Eq. (7).

A.5 Policy rewards and optimization

For a valid completed formula f and the current library $\mathcal{L}$, the shaped terminal reward is

$$\widetilde{r}_D(f \mid \mathcal{L}) = \alpha w(f,\mathcal{L}) q_D(f) + \eta_{\text{nov}} N(f,\mathcal{L}) - \eta_{\text{corr}} P_{\text{corr}}(f,\mathcal{L}) - \eta_{\text{dup}} P_{\text{dup}}(f,\mathcal{L}) - \eta_{\text{fam}} P_{\text{fam}}(f,\mathcal{L}). \tag{A.3}$$

Let $m(f,\mathcal{L})$ be the largest absolute correlation between the candidate signal and a retained library signal. Then $N = 1 - m$ and $P_{\text{corr}} = \max(0, m - \kappa)$, where $\kappa < 1$ is a correlation threshold. P_{dup} and P_{fam} indicate a previously seen formula and semantic family, respectively. The weight w equals one for direct quality rewards and $\max\{0, 1 - P_{\text{corr}}/(1-\kappa)\}$ for correlation-discounted rewards. The remaining coefficients are nonnegative reward scales. This reward is distinct from the raw signed target used to train the predictive models.

PPO returns combine construction rewards with this terminal reward. Updates use normalized advantages, a value loss, and entropy regularization, restricted to freshly sampled policy actions. Reward shaping affects policy optimization, while the predictive-model objectives continue to use the unshaped quality target $q_D(f_\tau)$.

B Planning Details

Path utility and cross-depth comparison. Equation (7) retains the cumulative path utility J_ϕ for beam pruning among candidates at the same depth. For comparison across depths, we normalize the quality and auxiliary components separately, since a length-k candidate contains $k+1$ quality predictions but only k auxiliary terms. Write $g_h = g_\phi(\widehat{z}_h, a_h)$, $p_h = p_\phi(\widehat{z}_h, a_h)$, and $v_k = v_\phi(\widehat{z}_k)$. For the candidate's actual length $1 \le k \le H$, define

$$\bar{J}_\phi(\xi \mid s_t) = (1+\gamma)\frac{\sum_{h=0}^{k-1}\gamma^h g_h + \gamma^k v_k}{\sum_{h=0}^{k}\gamma^h} + \beta\frac{\sum_{h=0}^{k-1}\gamma^h p_h}{\sum_{h=0}^{k-1}\gamma^h}. \tag{B.1}$$

The normalized score combines weighted averages of the quality and auxiliary predictions on the original one-step scale. Every intermediate assessment contributes to the comparison. If $g_h = v_k = q$ and $p_h = p$, then $\bar{J}_\phi = (1+\gamma)q + \beta p$ for every k, so equal predictions yield equal comparison scores across candidate lengths. The two scores serve complementary roles: J_ϕ guides beam expansion at each depth, while $\bar{J}_\phi$ places candidates from different depths on a common scale for action selection.

All generated depths contribute their normalized scores to (8); pruning limits further expansion but does not erase recorded candidates. Terminal candidates retain their scores at their actual length and are not expanded.

One-step consistency and reference comparison. For $k = 1$, $\bar{J}_\phi = J_\phi$ exactly. Thus, at $H = 1$, the rollout score remains

$$S^{(1)}_{\text{roll}}(s_t, a) = g_\phi(z_\phi(s_t), a) + \beta p_\phi(z_\phi(s_t), a) + \gamma v_\phi(F_\phi(z_\phi(s_t), a)). \tag{B.2}$$

Both the rollout and reference scores then contain one action-conditioned quality estimate, one successor-prefix quality estimate, and an auxiliary term. Their principal representational difference is that the rollout successor is predicted by F_ϕ, while the reference successor is encoded from its symbolic expression. At greater horizons, (8) and hence the discrepancy in (10) use $\bar{J}_\phi$. Its quality and auxiliary components retain total weights $1+\gamma$ and β, respectively, independent of candidate length; the reference retains weights $1+\gamma$ and β_{ref}. The discrepancy therefore relates the best discovered trajectory's normalized assessment to the direct assessment of its first action.

Let $\mathcal{A}_{\text{scored}}(s_t)$ be the legal first actions represented in $\mathcal{C}_t$. The admissible set is

$$\mathcal{A}_{\text{adm}}(s_t) = \{a \in \mathcal{A}_{\text{scored}}(s_t) : p_\phi(z_\phi(s_t), a) \ge p_{\min},\ \Delta(s_t, a) \le \delta_{\max}\}. \tag{B.3}$$

A disabled auxiliary threshold is represented by $p_{\min} = 0$; a disabled discrepancy threshold by $\delta_{\max} = +\infty$. If no scored action passes these filters, the reference score ranks the grammar-legal actions directly, without reapplying these filters. These thresholds affect action admissibility, whereas λ controls the score penalty among admissible actions.

C Experimental Details

C.1 Data and metrics

CSI300 and CSI500 use daily point-in-time constituent membership. The S&P 500 data use Yahoo prices and historical constituent snapshots reconstructed from Wikipedia (PIT-Approx). The prediction target is $y_d(i) = \mathrm{Close}_{d+20}(i)/\mathrm{Close}_d(i) - 1$. Daily Pearson and Spearman correlations use current constituents with finite signal–target pairs. Missing values and non-members are excluded.

IC and RIC are the absolute annual means of daily Pearson and Spearman correlations. ICIR and RICIR divide the corresponding annual means by their sample standard deviations across days, before taking absolute values. Each metric is averaged over 2023–2025 within a seed. Tables report the mean and sample standard deviation across seeds 0, 1, and 2.

C.2 Training and search settings

Table 1 uses the same annual walk-forward splits for AlphaDreamer and every baseline across all three markets. For test years 2023, 2024, and 2025, the nominal training intervals are 2010–2020, 2010–2021, and 2010–2022, respectively. The corresponding validation intervals are 2021–2022, 2022–2023, and 2023–2024. Every method is rerun on each split using only pre-test labels. AlphaGen uses its learned final pool for each split. AlphaQCM uses the QRDQN-generated factor pool. AlphaSAGE applies adaptive combination to its mined pool. AlphaForge dynamically selects and linearly combines factors from its mined zoo. Adaptive updates use only labels available at the prediction date. All interval endpoints respect the twenty-trading-day label maturity.

AlphaGen uses MaskablePPO for 250,000 steps with pool capacity 20. AlphaQCM uses QRDQN for 2,000,000 steps with pool capacity 20. AlphaSAGE mines for 10,000 episodes with pool capacity 50 and adaptively combines up to 20 factors. AlphaForge mines a 100-factor zoo and dynamically selects ten factors for combination at each prediction date. GP uses a population of 1,000 for 40 generations, ranks expressions by training IC, and combines the top 20 in an optimized factor pool.

AlphaDreamer uses OHLCV/VWAP features and arithmetic, cross-sectional, and time-series operators. On CSI300, the windows are $\{10, 20, 30, 40, 50\}$, with expression length at most 56 and depth at most nine. Up to 50 factors come from the validation-selected round. If it has $N < 50$ candidates, all N are retained. For each test year, each factor's direction is set by the sign of its mean validation IC and held fixed throughout testing.

C.3 Portfolio evaluation

Figure 3 plots $W_d = 1 + \sum_{u \le d} r_u$, where r_u is the selected basket's twenty-trading-day forward return less a fixed 0.1% cohort cost, divided by 20. CSI300 and CSI500 select the top 20% of stocks for a long-only basket; S&P 500 takes the top and bottom 10% for long and short positions. Curves start at one and show pointwise means across three seeds.

The S&P 500 panel combines forward-adjusted and backward-normalized prices under PIT-Approx membership. Signal directions are set without test returns. AR is a percentage; SR is dimensionless.

C.4 Planning comparison

Tables 2 and 3 use independent annual CSI300 models for seeds 0, 1, and 2. Each model trains for 16 rounds. Policy-only uses 1,024 scratch episodes per round; MPPP uses 512 scratch and 512 planner-guided episodes. Both allocate 16,384 generation attempts per year and seed. The planner has beam width 16, executes at most eight planned actions per episode, and uses $H \in \{1, \ldots, 9\}$. Planned actions are excluded from policy-gradient updates. Scratch and planned-completion updates use 16 and 12 PPO epochs, respectively; Policy-only uses 16 scratch epochs.

Both settings combine up to 50 factors from the validation-selected round. Evaluator calls count actual invocations. Unique valid formulas per 1,000 calls are calculated for each year and seed, averaged over years within seed, then summarized across seeds. Planning time is measured per decision.

Figure 4(a,b) uses eight fixed evaluator budgets from 1,500 to 5,000 calls. At each budget, completed snapshots are selected by validation ensemble IC, then validation-ranked factors form a portfolio of up to 50 formulas. Test absolute IC and RankIC are averaged over 2023–2025 within each seed before the three-seed mean and sample standard deviation are computed. Figure 4(c,d) compares full MPPP and S_{ref}-only selection using six paired seeds on the 2025 test split. Both variants follow the same training and factor-selection protocol; the ablation changes the planner's action score, not the evaluation metric.

D Additional Results

D.1 Planning horizon and prediction stability

Table 6 extends the horizon comparison to all four correlation metrics. $H = 1$, the main benchmark setting, has the highest mean IC, ICIR, RIC, and RICIR. Longer horizons do not consistently improve the results.

Table 6: Complete planning-horizon comparison on CSI300. Mean $\pm$ sample standard deviation across three seeds, after averaging absolute annual metrics over 2023–2025 within each seed. Bold denotes the highest mean in each column.

Configuration	IC (%) ↑	ICIR ↑	RIC (%) ↑	RICIR ↑
Policy-only	8.69 ± 0.60	0.482 ± 0.023	7.66 ± 0.57	0.483 ± 0.035
$H = 1$	$\mathbf{9.53} \pm 2.17$	$\mathbf{0.535} \pm 0.040$	$\mathbf{9.51} \pm 1.16$	$\mathbf{0.520} \pm 0.032$
$H = 2$	9.49 ± 1.51	0.514 ± 0.106	8.53 ± 2.53	0.464 ± 0.084
$H = 3$	7.39 ± 0.84	0.405 ± 0.017	7.00 ± 1.42	0.390 ± 0.024
$H = 4$	8.50 ± 2.05	0.496 ± 0.121	8.25 ± 1.34	0.472 ± 0.048
$H = 5$	8.29 ± 0.57	0.485 ± 0.044	7.01 ± 0.93	0.409 ± 0.009
$H = 6$	9.25 ± 0.78	0.502 ± 0.077	8.89 ± 1.13	0.463 ± 0.021
$H = 7$	8.03 ± 1.98	0.506 ± 0.187	7.15 ± 1.65	0.453 ± 0.162
$H = 8$	9.02 ± 1.69	0.521 ± 0.015	8.89 ± 2.51	0.505 ± 0.077
$H = 9$	8.95 ± 0.67	0.515 ± 0.027	7.88 ± 0.22	0.449 ± 0.030

D.2 Search efficiency

Table 7 partitions generation attempts by formula outcome. Canonical hashes identify first occurrences and repeats; only first occurrences invoke the evaluator. Planning raises the share of first-seen valid formulas from 6.32% to 9.51% while reducing evaluator calls from 50.18% to 37.48% of attempts. Its pooled yield is 25.38% valid formulas per call, versus 12.60% for Policy-only.

Table 7: Generation-attempt outcomes on CSI300. All entries are percentages of attempts, reported as mean $\pm$ sample standard deviation across three seeds after averaging within seed over 2023–2025. Categories are mutually exclusive.

Outcome	Policy-only	$H = 1$
No completed formula	49.82 ± 1.69	46.85 ± 0.67
First-seen valid	6.32 ± 2.33	9.51 ± 1.56
First-seen invalid	43.86 ± 1.32	27.97 ± 2.32
Repeated valid	0.00 ± 0.00	11.96 ± 1.22
Repeated invalid	0.00 ± 0.00	3.71 ± 1.15