TRADING STRATEGY OPTIMIZATION VIA TEXTUAL GRADIENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantitative trading strategy design aims to discover trading programs from historical data that remain effective in future markets, which can be viewed as a black-box program optimization problem. LLM-based textual gradients offer a promising approach by providing explicit optimization directions for iterative strategy refinement. However, directly applying textual gradients faces two challenges: (1) optimization is myopic, underutilizing experience from previous evaluations; and (2) aggregate backtest feedback overlooks temporal robustness, potentially favoring strategies that perform well only in specific market periods. To address these challenges, we propose TradeGrad, an experience-guided textual-gradient framework for robust trading strategy optimization. TradeGrad leverages accumulated optimization experience to estimate textual gradients and employs multi-scale revisions for both strategy exploration and refinement. It further introduces the Cross-Period Robust Objective (CPRO), which emphasizes performance in unfavorable historical periods to promote temporal robustness. Experiments on cross-sectional and time-series strategy design in Chinese A-share and U.S. equity markets show that TradeGrad achieves the best in-sample and out-of-sample performance across all four settings. Notably, its Chinese cross-sectional strategy achieves 27.99% annualized return, 12.19% maximum drawdown, and a Sharpe ratio of 1.63, approximately 68% higher than the CSI 300 benchmark. Further analyses validate the proposed components and show consistent improvements in both in-sample and out-of-sample performance throughout optimization. The code is available at `https://anonymous.4open.science/r/TradeGrad-7FC6`.

1 INTRODUCTION

Quantitative trading strategies are systematic decision rules that map market observations into trading decisions and play an important role in modern investment (Guo et al., 2024). In practice, a strategy is implemented as an executable program that integrates signal construction, entry and exit rules, position management, and risk control (Figure 1(a)). Trading strategy design can therefore be viewed as searching over a vast space of programs (Allen & Karjalainen, 1999). However, unlike conventional program search under a fixed evaluation setting, trading strategy design uses historical market data to discover programs for deployment in future markets, where conditions may differ substantially. The objective is thus to identify strategies whose effectiveness extends beyond historical backtests to unseen future market conditions (Bailey & López de Prado, 2014) (Figure 1(b)).

Traditionally, trading strategies were designed by quantitative researchers through a combination of financial expertise, domain intuition, and extensive empirical experimentation. The substantial expertise and effort required have motivated research into automated strategy design (Allen & Karjalainen, 1999; Lo et al., 2000). Recent advances in Large Language Models (LLMs) offer new opportunities to automate this process by combining code generation with evolutionary search (Novikov et al., 2025; Lange et al., 2026; Yun et al., 2025). Leveraging their prior knowledge of programming and finance, LLMs can generate various strategy code and thereby facilitate the exploration of the strategy space. Nevertheless, existing methods rely largely on stochastic mutations, providing limited guidance on how a strategy should be improved and thus requiring extensive search to discover effective refinements.

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.

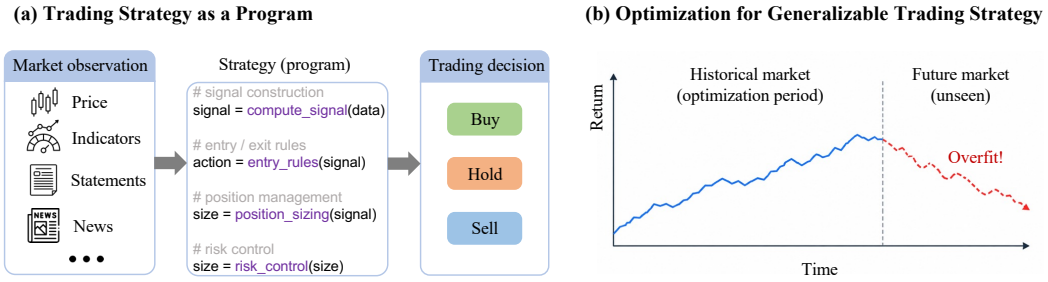


Figure 1: (a) Trading strategy as a program. (b) Optimization for Generalizable Trading Strategy.

Textual gradient optimization (Yuksekgonul et al., 2025) offers a more direct paradigm for iterative program refinement. Analogous to numerical gradients, it is defined as a natural-language optimization signal that diagnoses weaknesses in the current program and suggests actionable directions for revision. For example, if a momentum strategy performs well in trending markets but suffers large drawdowns during volatile reversals, a textual gradient may suggest normalizing momentum by recent volatility or requiring stronger trend consistency before entering a position. The LLM can then translate this semantic optimization direction into concrete modifications of the strategy program.

However, directly applying textual gradients to trading strategy optimization is unsatisfactory due to two key limitations. **(1) Textual-gradient optimization is myopic.** A gradient inferred from the current strategy captures only a narrow view of the strategy space. Earlier successes and failures are easily forgotten, causing redundant exploration and unstable optimization. Moreover, a fixed scale of revision cannot meet the varying needs of the search process: exploring alternative trading hypotheses requires structural changes, whereas refining promising strategies calls for localized edits. **(2) Aggregate backtest feedback overlooks temporal robustness.** Financial markets are non-stationary, and strategy performance can vary substantially across historical periods (Bailey & López de Prado, 2014; Bailey et al., 2017; Arian et al., 2024). An aggregate backtest score can be dominated by strong performance in favorable periods, even when the strategy performs poorly in others. Textual gradients derived from such feedback therefore direct optimization toward period-specific improvements that fail to generalize across market regimes. Effective optimization requires feedback that explicitly accounts for performance across different historical periods.

To address the above limitations, we propose **TradeGrad**, an experience-guided textual-gradient framework for robust trading strategy optimization, built around two central ideas. **(1) Experience-guided textual-gradient estimation** accumulates and reuses knowledge across strategy revisions. TradeGrad maintains a global optimization memory that distills successful and failure patterns from evaluated strategies. To estimate textual gradients, it samples archived strategies with their code and backtest results, providing comparative evidence for effective modifications. TradeGrad further employs multi-scale revisions: structural revisions explore alternative trading hypotheses, while localized code edits refine promising strategies. **(2) Cross-Period Robust Objective (CPRO)** promotes temporal robustness by evaluating strategies across calendar-year windows. Within each window, CPRO computes a composite score that integrates annualized return, maximum drawdown, and a one-sided penalty for insufficient trading activity. Inspired by Conditional Value-at-Risk (CVaR) (Rockafellar & Uryasev, 2002), CPRO aggregates the lower tail of these window-level scores, emphasizing unfavorable periods rather than allowing a few favorable periods to dominate.

We evaluate TradeGrad on cross-sectional and time-series strategy design in both Chinese A-share and U.S. equity markets. Extensive experimental results demonstrate that TradeGrad not only consistently achieves the highest in-sample (IS) and out-of-sample (OOS) scores among the evaluated strategy-design methods across all four settings but also maintains favorable risk–return trade-offs. Notably, in the Chinese market, its cross-sectional strategy achieves an annualized return of 27.99% with a maximum drawdown of only 12.19% and a Sharpe ratio of 1.63, outperforming the CSI 300 index by about 68% in Sharpe ratio. Ablation studies confirm that all proposed components are essential to TradeGrad's optimization performance. Optimization trajectory analysis shows that TradeGrad consistently improves both IS and OOS performance throughout the optimization, whereas compared methods tend to plateau or deteriorate.

The key contributions of this work are summarized as follows:

- We formulate trading strategy design as a black-box program optimization problem, and identify two limitations of applying textual-gradient optimization in this setting: myopic optimization that makes insufficient use of accumulated search experience, and aggregate backtest feedback that overlooks temporal robustness.
- We propose TradeGrad, an experience-guided textual-gradient framework for robust trading strategy optimization. By combining global optimization memory, sampling-based gradient estimation, and multi-scale strategy revisions, TradeGrad enables more efficient strategy optimization.
- TradeGrad introduces the Cross-Period Robust Objective that evaluates strategies across multiple calendar-year windows and emphasizes lower-tail performance, thereby promoting improvements that remain effective across market periods and generalize to unseen markets.
- Extensive experiments on cross-sectional and time-series strategy design in Chinese A-share and U.S. equity markets demonstrate that TradeGrad achieves the best IS and OOS performance among all evaluated methods and maintains favorable risk–return trade-offs. Ablation studies, hyperparameter analyses, and optimization trajectory analysis provide further insights into TradeGrad's effectiveness.

2 Problem Formulation

2.1 Black-Box Optimization over Strategy Programs

We formulate automated trading strategy design as a black-box optimization problem over executable programs. Let Π denote the space of executable strategy programs, and let $\mathcal{D} = \{x_t\}_{t=1}^{T}$ denote a sequence of market observations, where x_t contains the information available at time t. A trading strategy is represented by an executable program $\pi \in \Pi$ that maps the observed market history to a trading decision $a_t = \pi(x_{\leq t})$. A strategy may jointly specify signal construction, entry and exit rules, position management, portfolio allocation, and risk control.

Given a strategy π and a market dataset $\mathcal{D}$, a predefined backtesting procedure produces a backtest result $\mathcal{B}(\pi; \mathcal{D})$, which contains performance statistics such as annualized return, maximum drawdown, Sharpe ratio, and trading frequency. The quality of a strategy is evaluated through an objective function $J(\mathcal{B}) = J(\pi; \mathcal{D})$. In general, strategy evaluation may involve multiple objectives reflecting different aspects of trading performance. For notational simplicity, we use a scalar J throughout the paper to represent the objective used by the optimizer, where multiple performance criteria can be aggregated into J when necessary.

The optimizer iteratively proposes new strategy programs based on previously evaluated candidates. Specifically, a generic black-box optimizer can be represented as a proposal operator $\mathcal{U}$, which generates the strategy at iteration k as $\pi_k = \mathcal{U}\left(\{(\pi_i, \mathcal{B}_i, J_i)\}_{i=0}^{k-1}\right)$, where $\mathcal{B}_i$ denotes the backtest result of strategy π_i, and J_i denotes its corresponding objective value.

2.2 Optimization for Generalizable Trading Strategy

A distinctive challenge in trading strategy optimization is that performance on the historical data available during optimization is not itself the ultimate target. The goal is to obtain a strategy that performs well on future and unseen markets. Let p_{future} denote the unknown distribution of future market data. Ideally, the strategy should maximize

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathcal{D} \sim p_{\text{future}}} \left[J(\pi; \mathcal{D}) \right]. \tag{1}$$

In practice, p_{future} is unavailable during strategy development. We therefore divide the available historical data into an IS period $\mathcal{D}_{\text{IS}}$ and an OOS period $\mathcal{D}_{\text{OOS}}$. During optimization, the optimizer can access only backtest evaluations on $\mathcal{D}_{\text{IS}}$, whereas $\mathcal{D}_{\text{OOS}}$ is held out from the optimization process and used only to evaluate whether the resulting strategies generalize to unseen market periods.

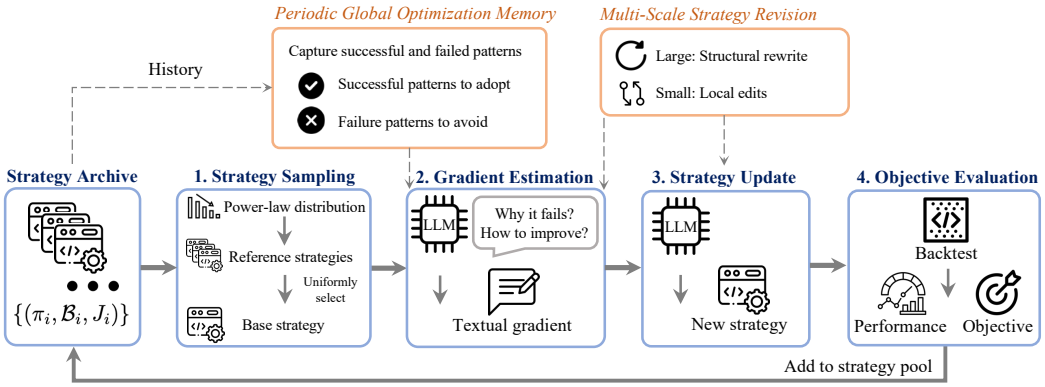


Figure 2: Overview of TradeGrad. TradeGrad iteratively samples archived strategies, estimates textual gradients from backtest feedback and optimization experience, updates strategies at varying revision scales, and evaluates and archives the resulting strategies.

3 METHOD

3.1 OVERVIEW

We propose TradeGrad, a textual-gradient optimization framework for automated trading strategy design, as illustrated in Figure 2 and Algorithm 1. TradeGrad maintains an archive of evaluated strategies and reuses them as optimization experience. At each iteration, it samples several strategies and selects one as the base strategy. An LLM compares the sampled strategies and their backtest results, together with experience summarized from previous iterations, to identify improvements to the base strategy. The resulting textual gradient guides its revision, ranging from structural rewrites to local edits. Finally, the new strategy is evaluated across multiple historical periods and added back to the archive, forming an iterative optimization loop.

3.2 EXPERIENCE-GUIDED TEXTUAL GRADIENT OPTIMIZATION

3.2.1 PERIODIC GLOBAL OPTIMIZATION MEMORY

As optimization proceeds, evaluated strategies and their backtest outcomes accumulate in the archive $\mathcal{A}_k$. To avoid repeatedly providing the growing archive to the LLM, TradeGrad maintains a persistent global memory $\mathcal{M}_k$ that periodically distills recent optimization experience into reusable successful and failure patterns. Formally, every H iterations, the preceding H experiences are summarized and consolidated with the previous memory:

$$\mathcal{M}_k = S_{\text{global}}\left(\{S_{\text{local}}(\pi_i, \mathcal{B}_i, J_i)\}_{i=k-H}^{k-1}, \mathcal{M}_{k-H}\right), \qquad k \bmod H = 0. \tag{2}$$

Here, S_{local} extracts reusable successful and failure patterns from an individual optimization experience, while S_{global} consolidates these summaries with the existing memory. Between refreshes, the latest memory is retained. The resulting $\mathcal{M}_k$ provides persistent experience that complements the strategies sampled from the current archive.

3.2.2 SAMPLING-BASED TEXTUAL GRADIENT ESTIMATION

At iteration k, strategies in $\mathcal{A}_{k-1}$ are ranked by their objective values J. We sample M reference strategies without replacement according to the power-law distribution

$$P(\pi_{(r)}) = \frac{r^{-\alpha_s}}{\sum_{j=1}^{|\mathcal{A}_{k-1}|} j^{-\alpha_s}}, \tag{3}$$

where $\pi_{(r)}$ denotes the strategy with rank r and $\alpha_s > 0$ controls the preference for highly ranked strategies. This produces a performance-biased reference set $\mathcal{S}_k = \{(\pi_k^{(j)}, \mathcal{B}_k^{(j)}, J_k^{(j)})\}_{j=1}^{M}$. We then uniformly select one strategy from $\mathcal{S}_k$ as the base strategy π_k^{base}. Thus, the reference context is biased toward stronger strategies, while the update target can originate from different quality levels.

Given the reference set $\mathcal{S}_k$, base strategy π_k^{base}, global memory $\mathcal{M}_k$, and revision scale η_k, the LLM estimates a textual gradient

$$\hat{g}_k = \text{LLM}\left(\mathcal{S}_k, \pi_k^{\text{base}}, \mathcal{M}_k, \eta_k\right). \tag{4}$$

The textual gradient identifies weaknesses of the base strategy and specifies actionable modifications by comparing strategy programs and their backtest outcomes.

3.2.3 Multi-Scale Strategy Revision

Program revisions can differ substantially in semantic magnitude, from restructuring the core strategy logic to modifying a local trading condition. To balance these behaviors, we sample a discrete revision scale

$$\eta_k = \begin{cases} \eta_{\text{large}}, & \text{with probability } p_{\text{large}}, \\ \eta_{\text{small}}, & \text{with probability } 1 - p_{\text{large}}. \end{cases} \tag{5}$$

The revision scale η_k specifies the intended scope of modification during textual gradient generation and subsequently determines how the program is revised. When $\eta_k = \eta_{\text{large}}$, the LLM is instructed to make substantial changes to the core trading logic, and the resulting strategy is implemented as a full rewrite of the base program. When $\eta_k = \eta_{\text{small}}$, the LLM focuses on localized improvements while preserving the overall strategy structure, and the revision is implemented through diff-based replacements of selected code segments. Therefore, large-scale revisions promote structural exploration, while small-scale revisions enable targeted refinement.

3.3 Cross-Period Robust Objective

3.3.1 Cross-Period Evaluation

Aggregate backtest performance may favor strategies whose gains are concentrated in a few favorable periods. Therefore, we evaluate each strategy across multiple historical windows. Specifically, we partition the in-sample data into N non-overlapping historical windows, $\mathcal{D}_{\text{IS}} = \left\{\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(N)}\right\}$. For each window, annualized return $AR_n(\pi)$ and maximum drawdown $MDD_n(\pi)$ are transformed into normalized scores:

$$s_{\text{AR}}^{(n)}(\pi) = \sigma(\lambda_{\text{AR}}(AR_n(\pi) - r_{\text{ref}})), \tag{6}$$

$$s_{\text{MDD}}^{(n)}(\pi) = \sigma(-\lambda_{\text{MDD}}(MDD_n(\pi) - d_{\text{ref}})), \tag{7}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function, λ_{AR} and λ_{MDD} control the sensitivities, and r_{ref} and d_{ref} are reference levels for return and drawdown, respectively. This transformation maps the two metrics to a common bounded scale while favoring higher returns and lower drawdowns.

We combine the normalized scores with a one-sided trading-frequency gate:

$$s_n(\pi) = 100 \times \left(s_{\text{AR}}^{(n)}(\pi)\right)^{\alpha} \left(s_{\text{MDD}}^{(n)}(\pi)\right)^{1-\alpha} \left[\min\left(1, \frac{\tau_n(\pi)}{\tau_0}\right)\right]^{\gamma}, \tag{8}$$

where α controls the trade-off between return and drawdown, $\tau_n(\pi)$ is the trading frequency, τ_0 is the desired minimum frequency, and γ controls the penalty strength. The gate penalizes insufficient trading activity without rewarding excessive trading beyond τ_0.

3.3.2 Lower-Tail Robust Objective

Inspired by Conditional Value-at-Risk (CVaR) (Rockafellar & Uryasev, 2002), we focus on the lower tail of window-level performance to emphasize unfavorable historical periods. Specifically, we sort the window scores as $s_{[1]}(\pi) \le \cdots \le s_{[N]}(\pi)$ and optimize the mean of the lowest 30%:

$$J(\pi; \mathcal{D}_{\text{IS}}) = \frac{1}{q} \sum_{n=1}^{q} s_{[n]}(\pi), \qquad q = \lceil 0.3N \rceil. \tag{9}$$

We refer to this objective as the Cross-Period Robust Objective (CPRO). By emphasizing lower-tail performance, CPRO prevents strong results in favorable periods from fully compensating for poor performance elsewhere, thereby promoting robustness across market periods.

After generating π_k, TradeGrad evaluates it using Equation (9) and adds the strategy and its evaluation to the archive, which provides experience for subsequent textual-gradient updates.

4 EXPERIMENTS

In this section, we conduct extensive experiments to answer the following research questions: **RQ1**: Does TradeGrad discover better trading strategies than existing methods? **RQ2**: How do the proposed mechanisms contribute to performance? **RQ3**: How do different hyperparameters affect the performance? **RQ4**: How does the optimization evolve throughout the search process?

4.1 EXPERIMENTAL SETUP

Markets and Evaluation. We evaluate TradeGrad on two representative markets: the Chinese A-share market and the U.S. stock market. For both markets, we use daily market data from 2018 to 2025, chronologically split into an IS optimization period and an OOS evaluation period with a ratio of 3:1. We consider two representative strategy-design settings. The first is the cross-sectional setting, where strategies trade individual stocks and construct portfolios by comparing assets at the same point in time. We use the point-in-time constituents of the CSI 300 for the Chinese market and the S&P 500 for the U.S. market. The second is the time-series setting, where strategies trade the corresponding market index based on its historical temporal dynamics. We evaluate strategies using IS Score (the value of the CPRO), OOS Score (the value of the aggregate objective), Annualized Return (AR), Maximum Drawdown (MDD), and Sharpe Ratio (SR). More implementation details are provided in Appendix A.2.

Baselines. We compare TradeGrad against four groups of baselines: **(1) Market index.** We report the corresponding market index as a passive benchmark for each market. **(2) Classical strategies.** We consider Momentum (Jegadeesh & Titman, 1993), Reversal (De Bondt & Thaler, 1985), Volatility (Ang et al., 2006), and Volume (Gervais et al., 2001) for cross-sectional trading, and Grid Trading (Rundo et al., 2019), Moving Average (MA) (Brock et al., 1992), Channel Breakout (Brock et al., 1992), and Bollinger Bands (Bollinger, 2002) for time-series trading. **(3) Evolutionary program search.** We consider two representative baselines. The first is the Island Algorithm, which combines LLM-driven code evolution with an island-based evolutionary framework and underlies general-purpose program evolution methods such as AlphaEvolve (Novikov et al., 2025) and ShinkaEvolve (Lange et al., 2026). The second is QuantEvolve (Yun et al., 2025), a domain-specific extension that incorporates financial guidance to tailor evolutionary search to quantitative trading. **(4) Textual-gradient optimization.** We use TextGrad (Yuksekgonul et al., 2025), which iteratively updates a single strategy according to textual gradients derived from its evaluation feedback.

4.2 OVERALL PERFORMANCE (RQ1)

We first answer RQ1 by evaluating the IS and OOS performance of the designed strategies. From the results shown in Table 1, we have the following findings.

Classical strategies remain competitive, while automated baselines struggle with cross-sectional trading. Volatility ranks second in OOS Score on CSI 300 cross-sectional trading, and Grid outperforms all automated baselines on CSI 300 time-series trading. In contrast, Island Algorithm, QuantEvolve, and TextGrad incur OOS losses in both cross-sectional markets. These automated baselines fare better in time-series trading, all achieving positive OOS returns in both markets, with QuantEvolve ranking second in OOS Score on S&P 500.

TradeGrad combines effective IS optimization with strong OOS generalization. TradeGrad achieves the highest IS and OOS Scores among all compared strategy-design methods in all four settings. This alignment distinguishes it from baselines whose historical success fails to carry over: For example, Island Algorithm ranks second in IS Score on S&P 500 cross-sectional trading but nearly last in OOS Score. TradeGrad's consistently leading performance on both splits supports its ability to discover strong historical strategies whose advantage persists on unseen data.

TradeGrad achieves favorable OOS risk–return trade-offs. TradeGrad obtains the highest OOS Sharpe ratio among strategy-design methods in all four settings. On CSI 300 cross-sectional trading, it achieves the highest AR of 27.99% with MDD of 12.19%. On S&P 500 cross-sectional trading, it delivers a positive AR of 6.05% with the lowest MDD of 3.93%. Its advantage also extends beyond maximizing returns: on S&P 500 time-series trading, TradeGrad earns a lower AR than MA but reduces MDD from 33.54% to 7.79% and increases SR from 0.88 to 1.32.

Table 1: Overall performance comparison. Results are averaged over three runs; standard deviations are reported in Appendix A.3. Best and second-best results are bolded and underlined, respectively.

		CSI 300					S&P 500				
Setting	Method	IS Score ↑	OOS Score ↑	AR ↑	MDD ↓	SR ↑	IS Score ↑	OOS Score ↑	AR ↑	MDD ↓	SR ↑
Market Index Reference		1.63	45.01	17.69	15.66	0.97	5.44	39.38	20.76	18.90	1.26
Cross-Sectional Strategies											
Classical	Momentum	0.52	9.19	5.76	28.49	0.35	9.02	2.37	-15.35	31.13	-1.58
	Reversal	0.28	14.51	17.80	28.59	0.74	5.12	26.76	0.56	11.59	0.11
	Volatility	7.56	60.69	18.59	**8.74**	1.41	4.41	6.93	-9.38	23.79	-0.79
	Volume	1.51	16.73	6.92	23.03	0.41	13.62	2.43	-16.08	30.54	-2.57
Evolution	Island Algorithm	0.23	2.52	-12.74	31.87	-0.60	16.84	0.60	-22.53	40.16	-4.22
	QuantEvolve	3.03	4.42	-5.59	30.14	-0.15	0.34	0.16	-26.75	49.90	-2.30
Gradient	TextGrad	0.09	4.40	-8.08	28.94	-0.40	1.02	3.51	-14.92	27.63	-2.62
	TradeGrad	**10.69**	**65.65**	**27.99**	12.19	**1.63**	**21.36**	**40.38**	**6.05**	**3.93**	**1.07**
Time-Series Strategies											
Classical	Grid	4.32	35.52	6.34	11.30	0.73	7.03	7.59	-11.39	21.66	-1.21
	MA	12.48	17.55	-0.22	18.52	0.04	0.67	10.08	**24.09**	33.54	0.88
	Breakout	25.69	28.11	1.13	10.96	0.17	4.60	30.42	2.64	10.77	0.36
	Bollinger	22.81	13.63	-7.89	16.50	-1.36	5.13	3.80	-14.37	27.13	-1.28
Evolution	Island Algorithm	26.04	30.08	3.50	12.26	0.38	14.22	36.02	11.57	15.77	1.06
	QuantEvolve	9.73	24.10	2.94	16.14	0.30	20.18	45.40	9.64	7.10	1.09
Gradient	TextGrad	0.89	29.48	0.05	**6.49**	0.03	0.65	32.93	2.23	**6.18**	0.34
	TradeGrad	**29.00**	**41.54**	**8.21**	8.59	**0.87**	**23.24**	**50.08**	12.39	7.79	**1.32**

Table 2: Ablation of experience-guided textual-gradient estimation. Results are mean ± standard deviation over three runs. Best and second-best means are bolded and underlined, respectively.

Method	IS Score ↑	OOS Score ↑	AR ↑	MDD ↓	SR ↑
Full Method	$\mathbf{10.69}_{\pm 1.06}$	$\mathbf{65.65}_{\pm 12.15}$	$\mathbf{27.99}_{\pm 4.19}$	$12.19_{\pm 4.65}$	$\mathbf{1.63}_{\pm 0.17}$
w/o Memory	$4.56_{\pm 0.42}$	$35.62_{\pm 7.07}$	$11.78_{\pm 1.73}$	$16.09_{\pm 5.97}$	$0.80_{\pm 0.08}$
w/o Sampling	$\underline{8.32}_{\pm 0.85}$	$46.72_{\pm 10.68}$	$12.98_{\pm 2.03}$	$\underline{11.35}_{\pm 4.38}$	$0.94_{\pm 0.10}$
w/o MSR (Diff)	$5.79_{\pm 0.54}$	$46.88_{\pm 12.10}$	$14.87_{\pm 1.14}$	$13.04_{\pm 5.14}$	$0.96_{\pm 0.10}$
w/o MSR (Rewrite)	$7.42_{\pm 0.77}$	$\underline{55.74}_{\pm 10.01}$	$\underline{15.98}_{\pm 2.39}$	$\mathbf{8.86}_{\pm 3.48}$	$\underline{0.98}_{\pm 0.11}$

4.3 Ablation Study (RQ2)

4.3.1 Ablation of Experience-Guided Textual-Gradient Estimation

To answer RQ2, we first ablate the three textual-gradient optimization mechanisms in Table 2: periodic global memory, sampling-based gradient estimation, and multi-scale revision (MSR). Under the same CPRO and evaluation budget, we remove memory or sampling, and replace MSR with Diff-only or Rewrite-only updates.

All three optimization mechanisms contribute to the final performance. Removing any proposed mechanism decreases both IS and OOS Scores. The full method achieves an OOS Score of 65.65, while removing sampling or memory and replacing MSR with a single revision mode all lead to substantial degradation. This consistent decline indicates that the three mechanisms provide complementary benefits to textual-gradient optimization.

Global memory makes the largest contribution to optimization. Removing global memory causes the largest performance drop among all ablations, reducing the OOS Score from 65.65 to 35.62. This suggests that retaining optimization experience across iterations is critical for guiding textual-gradient updates toward strategies that generalize beyond the optimization period.

MSR balances local refinement and broader exploration. Both Diff-only and Rewrite-only revisions underperform the full method, with Rewrite-only performing better than Diff-only. This suggests that broader revisions help escape weak search regions, while combining them with fine-grained modifications enables MSR to balance exploration and refinement more effectively.

Table 3: Ablation of the CPRO. IS OV reports each variant's own objective value and is not directly comparable across rows. Results are mean ± standard deviation over three runs. Best and second-best means are bolded and underlined, respectively.

Objective	IS OV ↑	IS Score ↑	OOS Score ↑	AR ↑	MDD ↓	SR ↑
CPRO	$10.69_{\pm 1.06}$	$\mathbf{10.69}_{\pm 1.06}$	$\mathbf{65.65}_{\pm 12.15}$	$\mathbf{27.99}_{\pm 4.19}$	$\mathbf{12.19}_{\pm 4.65}$	$\mathbf{1.63}_{\pm 0.17}$
CPRO w/o Freq Gate	$10.99_{\pm 1.07}$	$\underline{8.10}_{\pm 0.79}$	$\underline{40.24}_{\pm 8.68}$	$10.43_{\pm 1.51}$	$\underline{12.66}_{\pm 4.81}$	$0.73_{\pm 0.07}$
Mean-Period Objective	$37.72_{\pm 4.18}$	$4.40_{\pm 0.40}$	$23.37_{\pm 5.06}$	$6.08_{\pm 0.93}$	$18.70_{\pm 7.13}$	$0.44_{\pm 0.05}$
Aggregate Objective	$5.11_{\pm 0.56}$	$0.74_{\pm 0.07}$	$35.63_{\pm 7.64}$	$\underline{13.12}_{\pm 1.96}$	$16.89_{\pm 6.38}$	$\underline{0.77}_{\pm 0.09}$

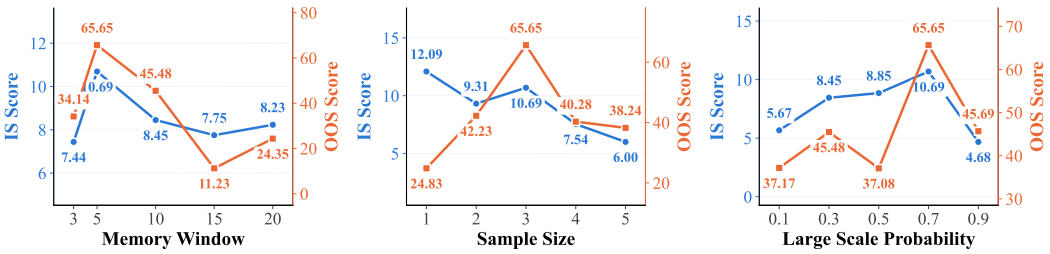


Figure 3: Effect of different hyperparameters: Memory window for summarizing optimization experience, sample size for textual-gradient estimation, and probability of large-scale revision.

4.3.2 Ablation of the Cross-Period Robust Objective

We further examine the effect of the optimization objective on OOS generalization in Table 3, comparing CPRO with variants that remove the frequency gate, average performance across periods, or evaluate performance over the aggregated optimization period.

The trading-frequency gate improves both IS performance and OOS generalization. Removing the frequency gate reduces the standardized IS Score from 10.69 to 8.10 and the OOS Score from 65.65 to 40.24. It also lowers OOS profitability and increases drawdown, with SR declining from 1.63 to 0.73. These results support the gate's role in guiding search toward strategies that perform better under the common evaluation criteria and retain their advantage out of sample.

Emphasizing weak periods is more effective than optimizing mean or aggregate performance. CPRO achieves the highest standardized IS Score, compared with 4.40 for the Mean-Period Objective and 0.74 for the Aggregate Objective. Its advantage extends to OOS evaluation, where it achieves a Score of 65.65, versus 23.37 and 35.63, respectively, together with higher AR and SR and lower MDD. These results support emphasizing the lower tail of historical performance: under the same search procedure and budget, this objective produces strategies with stronger performance under both the common IS criterion and held-out OOS evaluation.

4.4 Hyperparameter Analysis (RQ3)

Figure 3 presents a sensitivity analysis of three key hyperparameters, varying one at a time while keeping the others fixed. The analysis is intended to characterize the sensitivity of TradeGrad and to examine whether its behavior is consistent across the optimization and held-out periods.

Global memory exhibits a non-monotonic sensitivity to its temporal horizon. Performance improves as the memory window increases from 3 to 5, but generally declines with longer windows. One possible explanation is that longer windows may incorporate stale or heterogeneous experience that weakens subsequent updates.

The sample size shows different IS and OOS trends. A single sample achieves the highest IS Score, whereas a sample size of 3 shows stronger OOS performance. Further increasing the sample size degrades both metrics, suggesting that additional references may provide useful comparative evidence but can also introduce irrelevant or conflicting information.

MSR benefits from a mixture of revision scales. OOS performance generally improves as the probability of large revisions increases, but drops substantially at $p_{\text{large}} = 0.9$. Together with the MSR ablation, this pattern is consistent with the motivation for combining structural exploration with localized refinement.

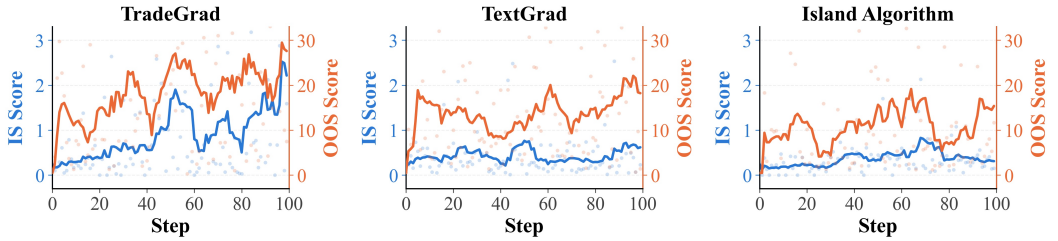


Figure 4: Optimization trajectories of different methods. Points denote the strategies generated at each optimization step, and curves show their 10-step moving averages (MA(10)).

4.5 Optimization Trajectories (RQ4)

To further understand the search dynamics, we track the IS and OOS Scores of the strategies generated throughout optimization. Figure 4 shows the strategy obtained at each step together with its 10-step moving average (MA(10)), which highlights the overall optimization trend.

TradeGrad achieves more sustained optimization and generalization than the baselines. TradeGrad shows a clear upward trend in both IS and OOS Scores throughout optimization, indicating that the search continues to discover stronger strategies and transfers these improvements out of sample. In contrast, TextGrad quickly plateaus after its initial gains, while Island Algorithm exhibits smaller and less persistent improvements with frequent regressions. Although all methods occasionally discover high-OOS strategies, the steadily rising trajectories of TradeGrad show that it more effectively accumulates optimization progress and translates it into sustained OOS gains. A case study illustrating the optimization process is provided in Appendix A.4.

5 Related Work

Automated Trading Strategy Design. Automated trading strategy design aims to discover executable trading rules from market data with limited human specification. Early work primarily used genetic algorithms to evolve trading rules across equity, foreign-exchange, and futures markets, jointly searching over strategy structures and parameters (Allen & Karjalainen, 1999; Neely & Weller, 1999; Wang, 2000; Potvin et al., 2004; Hu et al., 2015). Because flexible program search can overfit historical data, subsequent studies emphasize robustness through risk-aware objectives, transaction costs, out-of-sample evaluation, and evaluation across multiple market segments (Neely, 2003; Esfahanipour & Mousavi, 2011; Berutich et al., 2016). More recently, LLM-GA (Zhang et al., 2025), QuantEvolve (Yun et al., 2025), and ProFiT (Siper et al., 2026) use LLMs to generate and refine executable strategies, but remain within the evolutionary-search paradigm. In contrast, our work explores textual-gradient optimization for trading strategy design, with an explicit focus on out-of-sample robustness. More related work is discussed in Appendix A.5.

6 Conclusion

In this work, we formulate trading strategy design as a black-box program optimization problem and propose TradeGrad, an experience-guided textual-gradient framework for robust trading strategy optimization. TradeGrad leverages accumulated optimization experience to overcome the myopic nature of textual gradients and introduces the CPRO to promote temporal robustness across different market periods. Experiments on cross-sectional and time-series strategy design in both Chinese A-share and U.S. equity markets show that TradeGrad achieves strong in-sample and out-of-sample performance while maintaining favorable risk–return trade-offs, with further analyses validating the effectiveness of its key components. Future work can extend TradeGrad to broader asset classes and higher-frequency settings, incorporate richer market information into textual-gradient estimation, and investigate robustness under more diverse forms of distribution shift. More broadly, TradeGrad provides a promising direction for LLM-based program optimization in settings where solutions optimized on historical data need to generalize to unseen environments.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for generating synthetic datasets, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, interpreting results. Providing critical ingredients for proving mathematical claims and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, drafting parts of a research paper, summarizing or analysing existing literature, sourcing/searching for information, editing a research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. In particular, all literature identified or listed with the assistance of generative AI tools was independently checked by the authors for accuracy and relevance. LLM-generated code was reviewed, tested, and modified by the authors to verify its correctness and ensure that it implemented the intended functionality. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our work. The source code is available at `https://anonymous.4open.science/r/TradeGrad-7FC6`. All datasets used in this work are publicly accessible, with their sources and experimental splits described in Section 4.1. We provide detailed implementation and evaluation settings in Appendix A.2, including the evaluation metrics, TradeGrad hyperparameters and CPRO configuration, accessible market observations, optimization protocol, backtesting assumptions, and evaluation scope. Together, the released code, publicly available datasets, and documented experimental settings provide the necessary resources for reproducing the reported experiments and results.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999.

Andrew Ang, Robert J Hodrick, Yuhang Xing, and Xiaoyan Zhang. The cross-section of volatility and expected returns. *The Journal of finance*, 61(1):259–299, 2006.

Hamid Arian, Daniel Norouzi Mobarekeh, and Luis Seco. Backtest overfitting in the machine learning era: A comparison of out-of-sample testing methods in a synthetic controlled environment. *Knowledge-Based Systems*, 305:112477, 2024.

David Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *The Journal of Computational Finance*, 20(4):39–69, 2017.

David H Bailey and Marcos López de Prado. The deflated Sharpe ratio: Correcting for selection bias, backtest overfitting and non-normality. *Journal of Portfolio Management*, 40(5):94–107, 2014.

José Manuel Berutich, Francisco López, Francisco Luna, and David Quintana. Robust technical trading strategies using GP for algorithmic portfolio selection. *Expert Systems with Applications*, 46:307–315, 2016.

John Bollinger. *Bollinger on Bollinger bands*. McGraw-Hill New York, 2002.

William Brock, Josef Lakonishok, and Blake LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47(5):1731–1764, 1992.

Werner FM De Bondt and Richard Thaler. Does the stock market overreact? *The Journal of Finance*, 40(3):793–805, 1985.

Zixin Ding, Junyuan Hong, Zhan Shi, Tianhao Wang, Zinan Lin, Li Yin, Meng Liu, Zhangyang Wang, and Yuxin Chen. Scaling textual gradients via sampling-based momentum. In *ACM Conference on AI and Agentic Systems*, pp. 917–952, 2026.

Akbar Esfahanipour and Somayeh Mousavi. A genetic programming model to generate risk-adjusted technical trading rules in stock markets. *Expert Systems with Applications*, 38(7):8438–8445, 2011.

Simon Gervais, Ron Kaniel, and Dan H Mingelgrin. The high-volume return premium. *The Journal of Finance*, 56(3):877–919, 2001.

Jian Guo, Saizhuo Wang, Lionel M Ni, and Heung-Yeung Shum. Quant 4.0: Engineering quantitative investment with automated, explainable, and knowledge-driven artificial intelligence. *Frontiers of Information Technology & Electronic Engineering*, 25(11):1421–1445, 2024.

Yong Hu, Kang Liu, Xiangzhou Zhang, Lijun Su, EWT Ngai, and Mei Liu. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, 36:534–551, 2015.

Yeryung Jang and Jaekeol Choi. A prompt optimization system based on center-aware textual gradients. *Systems*, 13(9):748, 2025.

Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, 1993.

Robert Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, volume 2026, pp. 74026–74078, 2026.

Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *International Conference on Machine Learning*, 2024.

Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55 (4):1705–1765, 2000.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, et al. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, volume 2024, pp. 26516–26560, 2024.

Christopher J Neely. Risk-adjusted, ex ante, optimal technical trading rules in equity markets. *International Review of Economics & Finance*, 12(1):69–87, 2003.

Christopher J Neely and Paul A Weller. Technical trading rules in the European monetary system. *Journal of International Money and Finance*, 18(3):429–458, 1999.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Dengyun Peng, Yuhang Zhou, Qiguang Chen, Jinhao Liu, Jingjing Chen, Libo Qin, and Wanxiang Che. DLPO: Towards a robust, efficient, and generalizable prompt optimization framework from a deep-learning perspective. In *EMNLP (Findings)*, pp. 8311–8334, 2025.

Jean-Yves Potvin, Patrick Soriano, and Maxime Vallée. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023.

R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Francesco Rundo, Francesca Trenta, Agatino Luigi di Stallo, and Sebastiano Battiato. Grid trading system robot (GTSbot): A novel mathematical algorithm for trading FX market. *Applied Sciences*, 9(9):1796, 2019.

Matthew Siper, Ahmed Khalifa, Lisa Soros, Muhammad Nasir, Juyan Azhang, and Julian Togelius. ProFiT: Program search for financial trading. In *Genetic and Evolutionary Computation Conference*, pp. 727–734, 2026.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. In *AAAI Conference on Artificial Intelligence*, volume 39, pp. 25264–25272, 2025.

Niki Van Stein and Thomas Bäck. LlaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29 (2):331–345, 2024.

Jun Wang. Trading and hedging in S&P 500 spot and futures markets using genetic programming. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 20(10):911–942, 2000.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. *Neural Information Processing Systems*, 37:43571–43608, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Junhyeog Yun, Hyoun Jun Lee, and Insu Jeon. QuantEvolve: Automating quantitative strategy discovery through multi-agent evolutionary framework. *arXiv preprint arXiv:2510.18569*, 2025.

Di Zhang, Zhengyong Jiang, Qiong Ji, Hengyan Liu, Tianshi Wang, and Angelos Stefanidis. LLM-guided evolutionary strategy generation for quantitative trading. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3665–3670. IEEE, 2025.

A Appendix

A.1 Algorithm

The overall optimization procedure is summarized in Algorithm 1.

Algorithm 1 TradeGrad

Require: Initial strategy π_0, maximum optimization steps K
Ensure: Optimized strategy π^*

$\mathcal{B}_0 \leftarrow \mathcal{B}(\pi_0; \mathcal{D}_{\text{IS}})$
$J_0 \leftarrow J(\mathcal{B}_0)$
$\mathcal{A} \leftarrow \{(\pi_0, \mathcal{B}_0, J_0)\}$
$\mathcal{M} \leftarrow \emptyset$
for $k = 1, \ldots, K$ **do**
  if $k \bmod H = 0$ **then**
    $\mathcal{W} \leftarrow \{(\pi_i, \mathcal{B}_i, J_i)\}_{i=k-H}^{k-1}$
    $\mathcal{M} \leftarrow \textsc{UpdateMemory}(\mathcal{W}, \mathcal{M})$
  end if
  $\mathcal{S}_k \sim \textsc{RankBasedSample}(\mathcal{A})$
  $\pi_k^{\text{base}} \sim \textsc{UniformSample}(\mathcal{S}_k)$
  $\eta_k \leftarrow \textsc{SelectLR}()$
  $\hat{g}_k \leftarrow \textsc{TextGradient}(\mathcal{S}_k, \pi_k^{\text{base}}, \mathcal{M}, \eta_k)$
  $\pi_k \leftarrow \textsc{Update}(\pi_k^{\text{base}}, \hat{g}_k, \eta_k)$
  $\mathcal{B}_k \leftarrow \mathcal{B}(\pi_k; \mathcal{D}_{\text{IS}})$
  $J_k \leftarrow J(\mathcal{B}_k)$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(\pi_k, \mathcal{B}_k, J_k)\}$
end for
$\pi^* \leftarrow \arg\max_{(\pi, \mathcal{B}, J) \in \mathcal{A}} J$
return π^*

A.2 Implementation Details

A.2.1 Evaluation Metrics

For in-sample evaluation, to ensure a consistent basis for comparison, we report the value of the CPRO as the IS Score for all methods. For automatically searched strategies, the trading-frequency gate is included when computing the reported IS Score, whereas for classical strategies and the market-index reference, the frequency penalty is omitted.

For the optimization signal, following common practice in trading strategy optimization, the classical strategies and automated baselines are optimized or selected according to IS SR, except QuantEvolve, which follows the objective specified in its original method (Yun et al., 2025). TradeGrad instead uses CPRO for both optimization and strategy selection. Thus, the reported IS Score provides a common evaluation criterion while preserving the original optimization protocol of each method.

For out-of-sample evaluation, after strategy selection is completed using IS data, the selected strategy is evaluated once on the strictly held-out OOS period. We compute the OOS Score over the entire OOS period as

$$\text{OOS Score}(\pi) = 100 \left(s_{\text{AR}}^{\text{OOS}}(\pi)\right)^{\alpha} \left(s_{\text{MDD}}^{\text{OOS}}(\pi)\right)^{1-\alpha}, \tag{10}$$

where $s_{\text{AR}}^{\text{OOS}}(\pi)$ and $s_{\text{MDD}}^{\text{OOS}}(\pi)$ are computed from the annualized return and maximum drawdown over the entire OOS period using the transformations in Equation (7). Unlike IS Score, the OOS Score does not perform cross-period lower-tail aggregation or apply the trading-frequency gate.

We additionally report three standard OOS performance metrics:

- Annualized Return (AR): annualized growth of portfolio value.
- Maximum Drawdown (MDD): The largest peak-to-trough decline in cumulative returns.
- Sharpe Ratio (SR): Risk-adjusted return relative to return volatility.

A.2.2 Experimental Protocol

Market Observations. At each trading step, a strategy can access up to the preceding 252 trading days of historical market observations. For the Chinese A-share market, the available daily fields are

Table 4: Standard deviations of the main results over three independent runs.

	CSI 300					S&P 500				
Method	IS Score	OOS Score	AR	MDD	SR	IS Score	OOS Score	AR	MDD	SR
Cross-Sectional Strategies										
Island Algorithm	0.02	0.52	2.01	11.26	0.06	1.12	0.13	3.37	15.03	0.44
QuantEvolve	0.31	0.94	0.84	11.04	0.02	0.03	0.03	4.02	18.54	0.24
Vanilla TextGrad	0.01	0.97	1.21	11.50	0.04	0.10	0.74	2.23	10.09	0.27
Ours	1.06	12.15	4.19	4.65	0.17	2.67	8.39	0.94	1.50	0.11
Time-Series Strategies										
Island Algorithm	2.56	4.49	0.53	4.86	0.04	1.40	7.79	1.74	6.02	0.12
QuantEvolve	0.98	5.13	0.44	6.26	0.03	2.01	9.76	1.34	2.97	0.11
Vanilla TextGrad	0.09	6.45	0.01	2.48	0.01	0.06	7.10	0.33	2.36	0.04
Ours	2.90	8.86	1.32	3.24	0.08	2.32	10.70	1.86	2.71	0.14

open, high, low, close, volume, amount, and turnover. For the U.S. market, the available daily fields are open, high, low, close, and volume. No observations from the held-out OOS period are exposed to the optimizer during strategy optimization or selection.

Optimization Protocol. All LLM-based methods use Gemini-3-Flash[1]. To ensure comparability, all LLM-based methods use the same strategy representation, initial strategy information, accessible market observations, backtesting environment, and transaction-cost assumptions. Each method is given a maximum budget of 100 candidate strategies. We use the number of generated candidate strategies as the primary search budget because backtesting each candidate constitutes the common black-box evaluation operation across methods.

Backtesting Assumptions. All backtests start with an initial capital of $1{,}000{,}000$. We apply a proportional slippage rate of 0.05% and a commission rate of 0.05% per transaction. For the Chinese A-share market, we additionally apply a stamp tax rate of 0.1% on applicable transactions. All transaction costs are directly incorporated into strategy returns during backtesting. Chinese-market strategies are restricted to long-only positions, whereas U.S.-market strategies can take both long and short positions.

Evaluation Scope. The main experiments cover both markets and both strategy settings. All ablation studies, hyperparameter analyses, and optimization trajectory analyses are conducted in the CSI 300 cross-sectional setting. All reported results are averaged over three independent runs. For readability, Table 1 reports the mean values, while the corresponding standard deviations are provided in Appendix A.3.

A.3 Standard Deviations of Main Results

To assess the robustness of the experimental results, we repeat each stochastic method three times with different random seeds. Table 4 reports the standard deviations corresponding to the mean results in Table 1. Overall, the results exhibit consistent performance across independent runs.

A.4 Case Study

To understand how TradeGrad transforms trading programs during optimization, we trace the ancestral lineage of a representative final strategy. Figure 5 shows how the strategy logic and IS Score evolve from the initial program to the final solution.

TradeGrad evolves simple heuristics into substantially richer trading programs. The initial strategy contains only nine lines of logic, using three-day momentum computed solely from closing prices. Through successive revisions, TradeGrad expands both the information set and the decision structure. The final strategy contains 78 lines of logic and incorporates price, volume,

[1] https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf

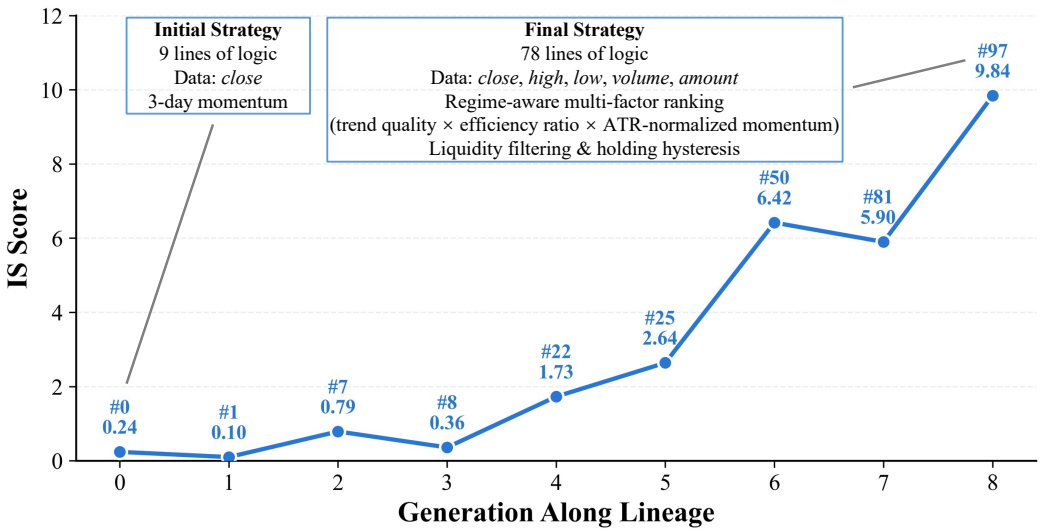


Figure 5: Evolution of a final strategy along its ancestral lineage. Labels indicate program IDs and IS Scores.

and trading-amount information. More importantly, it evolves from a single momentum signal into a regime-aware multi-factor strategy that combines trend quality, efficiency ratio, and ATR-normalized momentum with liquidity filtering and holding hysteresis. This evolution shows that TradeGrad searches beyond parameter tuning or local signal adjustment, progressively restructuring the executable trading logic.

Strategy evolution is non-monotonic but accumulates into substantial improvement. The ancestral lineage does not follow monotonic hill climbing. Several revisions temporarily reduce the IS Score, including $0.24 \rightarrow 0.10$, $0.79 \rightarrow 0.36$, and $6.42 \rightarrow 5.90$, yet these programs remain on the lineage leading to stronger descendants. Across eight generations, the strategy ultimately evolves from an IS Score of 0.24 to 9.84. This case illustrates the value of maintaining flexible program exploration: revisions that are not immediately beneficial can introduce new logic that becomes useful after subsequent refinement, allowing TradeGrad to reach solutions that would be difficult to obtain through strictly greedy updates.

A.5 More Related Work

LLM-based Program Optimization. LLMs have recently emerged as powerful operators for black-box program optimization, enabling search directly over executable code. FunSearch (Romera-Paredes et al., 2024) combines LLM-based program generation, automated evaluation, and evolutionary selection, a paradigm subsequently extended to reward functions, heuristics, and metaheuristics by Eureka (Ma et al., 2024), EoH (Liu et al., 2024), ReEvo (Ye et al., 2024), and LLaMEA (Van Stein & Bäck, 2024), and further scaled or improved in AlphaEvolve (Novikov et al., 2025) and ShinkaEvolve (Lange et al., 2026). These methods primarily optimize programs through evolutionary search, whereas textual gradients provide explicit semantic directions for program revision. Moreover, unlike typical program-optimization settings that optimize performance on a given evaluation problem, trading strategies must generalize from historical optimization data to unseen markets; our work studies textual-gradient program optimization under this additional generalization requirement.

Textual Gradient Optimization. Textual-gradient optimization uses natural-language feedback as an optimization signal for discrete and non-differentiable objects. Prior work has largely focused on prompt optimization: ProTeGi (Pryzant et al., 2023) uses LLM-generated critiques to guide revisions, OPRO (Yang et al., 2024) conditions optimization on previously evaluated solutions, GPO (Tang et al., 2025) studies gradient-inspired update directions and magnitudes, DLPO (Peng et al., 2025) introduces textual learning rates and decay, and GEPA (Agrawal et al., 2026) combines natural-language reflection with evolutionary search. Beyond prompts, TextGrad (Yuksekgonul

et al., 2025) formalizes textual gradients that can propagate through computation graphs to optimize code and other structured objects, while TSGD-M (Ding et al., 2026) and Center-Aware Textual Gradients (Jang & Choi, 2025) improve their stability and reliability. Applying this paradigm to trading introduces two additional challenges: limited exploration from single-trajectory updates and overfitting to historical backtests. We address these challenges through population-based textual-gradient optimization and a cross-period robust objective.