MATH 285J Final Report: Modular Agent with Reflective Search for Systematic Trading

Rigel Mummers[1] and Brendan Connelly[1]

[1]Department of Mathematics, University of California, Los Angeles, 90095, CA. Email(s): rigelmummers@ucla.edu, brendanconnelly@ucla.edu

Supervisor(s): Mihai Cucuringu

Abstract

Systematic trading research is a sequential search problem in which many candidate strategies are tried on the same historical data, making raw in-sample Sharpe optimization especially prone to backtest overfitting. We build a modular LLM/MCTS search framework over four strategy components (Universe, Signal, Processor, and Sizer) and test whether a robustness-aware reward $R(s)$—based on the Deflated Sharpe Ratio with regime-stability and capacity-cost penalties—selects strategies that generalize better than raw Sharpe. Overall, we found a negative result largely attributed to a lack of search diversity. Across a pooled body of 153 behaviorally distinct discovered strategies, no robustness objective out-predicts raw in-sample Sharpe on two out-of-sample windows. In repeated runs from the moving-average seed strategy, optimizing raw Sharpe produced higher held-out Sharpe than optimizing $R(s)$, although with only four repetitions the difference is not statistically conclusive. On a second seed (overnight-loser strategy), the picture is clearer: the DSR condition achieved held-out Sharpe $+0.537$ versus $+0.274$ for the Sharpe condition, despite the Sharpe condition's $12\times$ higher in-sample score—a result that supports the core hypothesis when sufficient seed variety is present. The deeper failure was that the search generated too many near-duplicate strategies: many evaluated nodes behaved like one effective strategy family. As a result, $R(s)$ had little meaningful variation to rank. Budget-sensitivity experiments further reveal that $R(s)$ conditions lock onto their best node within the first $\sim$30 evaluations and never improve, while Sharpe conditions keep discovering better nodes at higher budgets—suggesting the DSR penalty's exploitation pressure contributes to premature convergence. Additionally, the lesson pool—intended to give the search memory—appears to hinder rather than help: removing it consistently improves held-out performance at every budget level. The main contribution is then more diagnostic rather than promotional: we do not show that the method works, but we identify why it failed. The search produced too many similar strategies, some generated code changed the source without changing the portfolio, PBO looked better than it really was because the candidates were near-duplicates, and one-run LLM ablations gave unstable conclusions.

1 Introduction

Building a systematic trading strategy is a search problem: out of a huge space of possible signals, stock-universe filters, and position-sizing rules, we want the few combinations that actually work. A strategy that maximizes the in-sample Sharpe ratio often does so by fitting noise: Harvey et al. (2016) argue that conventional significance thresholds are too weak after large-scale factor search, and Bailey et al. (2014) show this is essentially unavoidable once enough strategies are tried on the same data—some will look good by chance alone. Standard automated searches make this worse by treating each candidate as an independent guess and discarding what they could learn from the ones that fail.

Our idea is to build robustness into the search objective itself rather than filtering for it afterward. We combine a Monte Carlo tree search over a modular strategy space, an LLM agent pipeline that proposes and critiques changes, and a reward $R(s)$ built on the Deflated Sharpe Ratio (Bailey and López de Prado, 2014)—a Sharpe ratio penalized for how many strategies were tried. The question we test is whether optimizing this robustness-aware reward, instead of the raw in-sample Sharpe, actually produce strategies that hold up better out of sample? Moreover, we inherently are testing the efficacy of Agentic AI in the kind of iterative workflow that is of rapidly growing importance and in the areas in which human analysts typically live.

2 Literature Review

2.1 Automated Search and Backtest Overfitting

Automated trading-rule discovery has a long history, including genetic algorithms over technical indicators (Allen and Karjalainen, 1999); more general automated model search methods, such as Bayesian hyper-parameter optimization (Bergstra et al., 2011), similarly evaluate many candidates by a scalar objective. In both cases, a central danger is treating each candidate score at face value without fully accounting for how many alternatives were tried. That omission is a core problem: Harvey et al. (2016) show that many reported factors do not survive multiplicity-adjusted significance thresholds, and Bailey et al. (2014) prove that enough trials on one sample will surface a strong-looking strategy under the null. The Probabilistic Sharpe Ratio of Bailey and López de Prado (2012) adjusts Sharpe-ratio inference for finite samples and non-normal returns, while the Deflated Sharpe Ratio (DSR) of Bailey and López de Prado (2014) adds an explicit correction for selection bias under multiple testing; the latter is the basis for the statistical part of our reward.

2.2 Modular Construction and Tree Search

Strategy performance is meaningfully separable across construction stages: Ciliberti and Gualdi (2018) show that normalization and weighting choices move Sharpe, turnover, and tails independently of the signal, and Bender and Wang (2016) find that bottom-up multi-factor construction captures interaction effects that combining single-factor portfolios misses. This motivates treating the module—universe, signal, processor, sizer—rather than the monolithic strategy as the unit of search and attribution. For the search itself we use UCT (Kocsis and Szepesvári, 2006), which applies an Upper Confidence Bound criterion to tree nodes with bounded regret and concentrates a limited evaluation budget in promising subtrees—the relevant property when each evaluation is an expensive backtest. A practical limitation of this workflow is reproducibility: LLM calls are stochastic, model versions will change and improve, and API costs limit the number of exact reruns. We therefore save generated strategies and re-score them under a fixed evaluation protocol rather than relying only on the original agent trajectory.

2.3 LLM Agents and Reflective Search

Our architecture follows recent work that combines LLMs with tree search and verbal memory: LATS (Zhou et al., 2024) uses an LLM as proposal generator, value estimator, and critic inside MCTS; Reflexion (Shinn et al., 2023) shows structured verbal feedback on past failures can improve an agent without weight updates; and the recent MARS working paper (Chen et al., 2026) studies MCTS with a persistent cross-run lesson pool. Together they motivate pairing semantic proposal generation, tree search, and reflective memory, which we adapt to systematic trading. Separately, Li et al. (2026) evaluate LLM-based investment strategies over longer horizons and broader universes, reinforcing the in-sample-versus-out-of-sample gap our objective targets.

2.4 Capacity Limits and Structured Modules

A backtest can also overstate performance because a strategy's edge shrinks as more capital trades on it; Cartea et al. (2025a) estimate this decay from a strategy's trading volume and concentration, motivating the capacity factor φ_{cost} of Section 5.2. Several results from our research group (Cartea et al., 2023a,b; Khelifa et al., 2024; Brutti et al., 2025; Cartea et al., 2025b) offer richer signal, universe, and sizing modules that could seed the search in future work; the present experiments use only four simple seeds.

3 Contributions

This paper's contribution is a negative but diagnostic result. In an LLM/MCTS strategy-search system, a robustness-aware objective $R(s)$—built on the Deflated Sharpe Ratio (Bailey and López de Prado, 2014) with regime-stability and cost-stress penalties—does *not* improve out-of-sample selection over raw in-sample Sharpe at the search diversity we attain. The central empirical finding is that the search collapses to very few effective strategy clusters, which makes the choice of reward largely secondary. Along the way we introduce practical diagnostics that expose failure modes ordinary backtest summaries hide: a return-correlation cluster count K_{eff}, the corresponding DSR trial input N_{eff}, position-fingerprint detection of behavioral no-ops, joint reporting of the Probability of Backtest Overfitting with K_{eff}, and an *objective horse-race* that scores any candidate objective by how well it predicts out-of-sample performance.

4 Data

4.1 Source and Coverage

We use the Center for Research in Security Prices (CRSP) daily stock file, which provides end-of-day prices, returns, shares outstanding, trading volume, and share and exchange codes for all securities listed on NYSE, AMEX, and NASDAQ. The full sample spans January 1990 to December 2024. The search uses 2000–2021 (in-sample walk-forward windows); the pre-2000 decade (1990–1999) and the post-2021 block (2022–2024) are held out and enter only the out-of-sample analysis. No held-out information enters node selection or objective computation.

4.2 Scale and Data Limitations

The empirical unit for strategy evaluation is the daily portfolio return. Each candidate strategy is scored over 22 annual walk-forward windows in 2000–2021, giving approximately 22×252 in-sample daily return observations after the warm-up requirements, and is then evaluated on two out-of-sample blocks: roughly 10×252 daily observations in 1990–1999 and 3×252 in 2022–2024. At the search level, the final objective horse-race uses 174 evaluated nodes and 153 behaviorally distinct strategies after deduplication by position fingerprint. The number of underlying stocks varies by date because the NYSE, common-share, price, and liquidity filters are point-in-time filters rather than a fixed asset list.

The main data limitations are deliberate design choices. We use daily CRSP data rather than intraday data, restrict the base universe to NYSE common shares, and model costs with simplified proportional and square-root impact assumptions. These choices make the experiments reproducible and reduce microstructure noise, but they also mean the results should be read as evidence about research-search behavior rather than as a claim that any discovered strategy is ready for live trading.

4.3 Universe Construction

On each trading day t, we first decide which stocks the strategy is allowed to trade. This set is the investable universe $\mathcal{U}_t$. We construct it from CRSP using only information that would have been available at the time. In particular, share code, exchange code, and SIC industry code are taken from CRSP event-history records, using the most recent record effective on or before date t, rather than from static header fields that may reflect later revisions. The liquidity filter is also lagged, so it only uses volume observed before the portfolio is formed.

A stock is included in $\mathcal{U}_t$ only if it passes four filters:

- **Exchange**: The stock must be listed on the NYSE (`EXCHCD` $= 1$). We exclude AMEX and NASDAQ stocks to keep the universe focused on larger, more liquid names and to reduce microstructure noise.
- **Share code**: The stock must be an ordinary common share (`SHRCD` $\in \{10, 11\}$). This removes ADRs, REITs, closed-end funds, preferred shares, and other securities that are not standard operating-company common stocks.
- **Liquidity**: The stock must have trailing 21-day average daily dollar volume above \$1 million, computed using only data available before date t. This removes very illiquid names whose simulated trades would be less reliable.
- **Price**: The stock must have a price above \$5 using the most recent price available before portfolio formation. This removes penny stocks, where measured returns are more likely to be distorted by bid-ask bounce, stale prices, and other microstructure effects.

Delisting returns. We include CRSP delisting returns to avoid overstating performance for stocks that disappear from the sample. Following Shumway (1997), we use the CRSP delisting return when available and impute a (-30%) return for missing performance-related delistings (codes 500–584). This adjustment is applied before computing strategy returns and backtest statistics.

4.4 Time Splits and Walk-Forward Windows

The in-sample period (2000–2021) is divided into $W = 22$ non-overlapping annual walk-forward windows of 252 trading days, each with a preceding 756-day (three-year) warm-up for signal computation; the robustness score $R(s)$ aggregates statistics across these windows (Section 5). The pre-2000 decade (1990–1999) and the post-2021 block (2022–2024) are held out from search and used only for the out-of-sample analysis of Section 7. Market regimes for ϕ_{stab} are defined by quintiles of the trailing 60-day realized volatility of the equal-weighted universe return.

5 Methodology

We describe the framework in five parts. First, we define the modular strategy representation and backtest engine (Section 5.1). Second, we define the robustness score $R(s)$ (Section 5.2). Third, we define the effective number of independent trials used inside that score (Section 5.3). Fourth, we describe the Monte Carlo tree search procedure (Section 5.4). Finally, we describe the LLM agent pipeline and lesson pool (Section 5.5).

5.1 Modular Strategy Representation

Each strategy s is built from four interchangeable pieces:

$$s = (\mathcal{M}_U, \mathcal{M}_S, \mathcal{M}_P, \mathcal{M}_Z).$$

Here $\mathcal{M}_U$ is the Universe module, $\mathcal{M}_S$ is the Signal module, $\mathcal{M}_P$ is the Processor module, and $\mathcal{M}_Z$ is the Sizer module. In the implementation, each module is a Python class with a fixed interface. This means that a signal module, for example, can be replaced without changing the universe, processor, or sizing code.

Universe. The Universe module $\mathcal{M}_U$ decides which stocks are eligible to be traded on day t. It maps the full CRSP panel to an investable set

$$\mathcal{U}_t \subseteq \mathcal{S},$$

where $\mathcal{S}$ is the set of all securities in the cleaned data store. This module contains the exchange, share-code, sector, price, and liquidity filters.

Signal. The Signal module $\mathcal{M}_S$ assigns each eligible stock a raw alpha score. For each security $i \in \mathcal{U}_t$, it produces

$$\alpha_{i,t} \in \mathbb{R}.$$

This is the part of the strategy that encodes the main economic idea, such as momentum, low volatility, or a moving-average rule.

Processor. The Processor module $\mathcal{M}_P$ turns the raw scores into scores that can be compared across stocks. It maps

$$\{\alpha_{i,t}\}_{i\in\mathcal{U}_t} \quad \text{to} \quad \{\tilde{\alpha}_{i,t}\}_{i\in\mathcal{U}_t}.$$

For example, the processor might cross-sectionally z-score the signal or convert it into ranks. This step matters because the raw signal values may not be on a useful scale for portfolio construction.

Sizer. The Sizer module $\mathcal{M}_Z$ converts the processed scores into portfolio weights:

$$\{\tilde{\alpha}_{i,t}\}_{i\in\mathcal{U}_t} \quad \mapsto \quad \{w_{i,t}\}_{i\in\mathcal{U}_t}.$$

The weights satisfy

$$\sum_i |w_{i,t}| = 1.$$

Standard sizing rules include signal-proportional weighting and volatility-parity weighting.

The daily portfolio return is

$$r_t = \sum_{i\in\mathcal{U}_t} w_{i,t-1}\, \tilde{r}_{i,t},$$

where $\tilde{r}_{i,t}$ is the delisting-adjusted return of stock i on day t. The weights $w_{i,t-1}$ are formed using information available at the close of day $t-1$, so the backtest does not use information from the return day itself.

Transaction costs are modeled as a proportional cost of $\kappa = 10$ basis points one-way per unit of turnover (the default `base_cost_bps=10.0` in `robustness/evaluator.py`). Turnover is defined as

$$\sum_i |w_{i,t} - w^+_{i,t-1}|,$$

where $w^{+}_{i,t-1}$ is the portfolio weight carried forward after price changes but before the new rebalance on day t.

5.2 Robustness Score

Raw in-sample Sharpe is a natural first objective, but it is too easy to fool in this setting. It does not account for the number of strategies tried, whether performance comes from only one market regime, or whether the strategy would survive more realistic trading costs. We therefore score each strategy s with a composite robustness reward:

$$R(s) = \mathrm{DSR}(s) \cdot \phi_{\mathrm{stab}}(s) \cdot \phi_{\mathrm{cost}}(s).$$

The three terms penalize different failure modes. DSR deflates the observed Sharpe for finite-sample noise, non-normal returns, and multiple testing. ϕ_{stab} penalizes strategies whose performance is concentrated in favorable volatility regimes. ϕ_{cost} penalizes strategies whose Sharpe disappears once trading costs and capacity are stressed. The product is a bounded search score, not a calibrated probability.

Deflated Sharpe Ratio. Following Bailey and López de Prado (2014), we use the Deflated Sharpe Ratio to adjust the observed Sharpe for the fact that the search evaluates many candidates:

$$\mathrm{DSR}(s) = \Phi\left(\frac{(\widehat{SR} - SR^{*})\sqrt{T-1}}{\sqrt{1 - \hat{\gamma}_3 \widehat{SR} + \frac{\hat{\gamma}_4 - 1}{4}\widehat{SR}^{2}}}\right).$$

Here $\widehat{SR}$ is the per-period Sharpe ratio computed from T daily returns, $\hat{\gamma}_3$ and $\hat{\gamma}_4$ are the sample skewness and kurtosis of the return series, and SR^{*} is the Sharpe level expected from the best strategy under the null after multiple trials. We compute this multiple-testing adjustment using the effective number of independent strategy clusters, N_{eff}, rather than the raw number of evaluated nodes. This matters because many generated strategies are near-duplicates.

During MCTS, the reward must be stationary: the same strategy should not receive a different score simply because it was found earlier or later in the run. For search, we therefore compute DSR using the fixed pre-registered budget $N_{\mathrm{budget}} = B = 100$. For final reporting, we also report the full N_{eff}-based DSR after deduplicating near-identical strategies.

Regime stability. To measure whether a strategy works across market environments, we split days into five volatility regimes using quintiles of trailing 60-day realized volatility. The quintile cutoffs are computed with information available before each walk-forward test period, so future volatility does not leak into the regime labels. Let $\widehat{SR}_k$ be the strategy's Sharpe in regime k, and let $\widehat{SR}_{(0.2)}$ be the 20th percentile of the regime-level Sharpes. We define

$$\phi_{\mathrm{stab}} = \sigma\left(\alpha_{\mathrm{stab}}\left(\frac{\widehat{SR}_{(0.2)}}{\max(|\overline{SR}|, \epsilon)} - 1\right)\right),$$

where $\sigma(x) = 1/(1+e^{-x})$, $\overline{SR}$ is the average Sharpe across regimes, $\epsilon = 10^{-6}$, and $\alpha_{\mathrm{stab}} = 3$. Strategies with negative average cross-regime Sharpe are assigned $\phi_{\mathrm{stab}} = 0$. Otherwise, the factor falls below $1/2$ when performance is concentrated in favorable regimes and rises above $1/2$ only when the lower-tail regime performance is at least as good as the average.

Capacity-aware cost stress. Finally, we penalize strategies that only work at very small scale. A fixed cost multiplier is too crude because trading costs generally rise with deployed capital (Cartea et al., 2025a).

For each strategy, we estimate net Sharpe along a capacity curve from $A_1 = \$1\text{M}$ to $A_L = \$100\text{M}$, using a square-root market-impact model. Let A^* be the largest AUM at which net Sharpe remains above $\tau_{SR} = 0.5$. We map this capacity estimate into a bounded penalty:

$$\phi_{\text{cost}} = \sigma\left(\alpha_{\text{cost}}\left(2\,\frac{\log A^* - \log A_1}{\log A_L - \log A_1} - 1\right)\right), \qquad \alpha_{\text{cost}} = 3.$$

A strategy that only works at the minimum AUM receives a score near 0.05, while one that remains viable at the maximum AUM receives a score near 0.95. We report A^* separately as a capacity diagnostic.

5.3 Effective Number of Independent Trials

The Deflated Sharpe Ratio needs a count of how many independent strategies were tried. The raw node count is not a good answer in our search tree. Many nodes are small variations of nearby nodes: TUNE changes parameters, IMPROVE swaps one module, and COMBINE copies a module from another strategy. These strategies can have very similar returns, so treating them as independent trials would overstate the amount of search we actually performed.

Following the clustering idea in López de Prado (2018), we estimate an effective trial count from return correlations. After all B nodes are evaluated, we compute the $B \times B$ correlation matrix of strategy returns. We then cluster strategies whose returns are highly correlated, using average-linkage hierarchical clustering and a correlation cutoff $\rho^* = 0.7$. Let K_{eff} be the number of resulting return clusters.

We use

$$N_{\text{eff}} = \max(K_{\text{eff}}, 2)$$

inside the DSR calculation. The floor at 2 is only a numerical safeguard, since the order statistic in the DSR formula is not well-defined for a single trial. In the results, we report K_{eff} next to the raw number of evaluated nodes. This makes the search diversity visible: $K_{\text{eff}} = 1$ means the search produced many variants, but they behaved like one effective strategy family. When $K_{\text{eff}} = 1$ the across-cluster Sharpe variance degenerates to zero; the implementation substitutes the variance of individual node daily Sharpe ratios within the single cluster as the estimate of $\text{Var}(\hat{\sigma})$ in the DSR formula (see `loop.py`, lines 358–371).

5.4 Monte Carlo Tree Search

We use Monte Carlo Tree Search to decide which strategy variants to evaluate next. Each node in the tree is a strategy, and each edge is an edit to that strategy. At each step, the algorithm chooses a node using the UCT rule (Kocsis and Szepesvári, 2006):

$$\text{UCT}(v) = \bar{R}(v) + c\sqrt{\frac{\ln N(v_{\text{parent}})}{N(v)}}, \qquad c = 0.7.$$

Here $\bar{R}(v)$ is the average robustness score of strategies evaluated below node v, and $N(v)$ is the number of times that node has been visited. The first term favors branches that have performed well. The second term gives less-visited branches an exploration bonus.

Each expansion uses one of four actions:

- **Draft**: propose a new strategy idea with no module constraint, using the LLM pipeline.
- **Improve**: replace the weakest module in the current strategy, using the LLM pipeline.
- **Tune**: randomly perturb numeric parameters by ±20%, without an LLM call.

- **Combine**: copy one module from the highest-scoring strategy into the current strategy.

After the new strategy is backtested, its score is propagated back up the tree. We use average rather than maximum backup because one strong child does not necessarily mean the whole branch is promising. Since each evaluation requires a full backtest, and sometimes an LLM call, we set $c = 0.7$ to lean somewhat toward exploitation while still preserving exploration.

5.5 LLM Agent Pipeline and Lesson Pool

The DRAFT and IMPROVE actions use a five-agent LLM pipeline before a strategy is backtested (Table 1). The purpose of the pipeline is to make each proposed change look more like a small research step than a raw code generation attempt: one agent proposes the idea, others check novelty and economic plausibility, another writes the code, and a final agent records what was learned.

Agent	Role
Idea	proposes a module change with a mechanism and economic rationale
Skeptic	checks whether the proposal is meaningfully new
Economic	rejects look-ahead or mechanism-free proposals
Coding	implements the module and tests it in a sandbox
Distill	records a positive or negative lesson from the result

Table 1: The five-agent proposal pipeline.

Two implementation details matter for the results. First, the Coding Agent runs each generated module in a restricted sandbox using only `numpy` and `pandas`, and retries up to three times when the code fails. Second, the Distill Agent records negative lessons as well as positive ones. A strategy with positive Sharpe but low $R(s)$, for example, is not treated as a success; it becomes a warning about a mechanism that looked profitable but failed the robustness test.

The lesson pool is a persistent JSON store shared across runs. Lessons are indexed by module type and mechanism tag, such as `signal:momentum`. When the Idea Agent proposes a change to a module, it receives a small number of relevant positive lessons and negative warnings from prior evaluations. This gives the search memory across MCTS episodes without changing the underlying language model.

Seeded baseline strategies. Each run starts from one of four simple baseline strategies rather than from a blank slate: 12–1 cross-sectional momentum (Jegadeesh and Titman, 1993), moving-average trend (Moskowitz et al., 2012), overnight reversal (Lou et al., 2019), and low volatility (Baker et al., 2011; Frazzini and Pedersen, 2014). These seeds share the same basic universe, processor, and sizer, and differ mainly in the Signal module. Starting from known strategy families keeps the search in economically interpretable regions and gives each discovered strategy a natural baseline for comparison.

Richer modules from related work—including correlation-clustering and lead-lag signals (Cartea et al., 2023a,b), hierarchical universes (Khelifa et al., 2024), and robust or learned-characteristic portfolio construction (Brutti et al., 2025; Cartea et al., 2025b)—are natural extensions, but they are not implemented in the present experiments.

5.6 Evaluation Protocol

We evaluate strategies on three time blocks. The search itself uses 2000–2021. The period 2022–2024 is not used inside the search objective, lesson pool, prompts, or hyperparameter selection, but because we

inspected it during development, we treat it as an *inspected held-out diagnostic* rather than a fully sealed test set. This cuts in favor of, not against, the paper's main negative finding: inspection during development would if anything bias results toward making the robustness objective look better, so the fact that $R(s)$ still fails to beat raw Sharpe on this block is a conservative result, not an inflated one. Within each run, the best node is selected using only the run's in-sample objective over 2000–2021, either $R(s)$ or raw Sharpe, and is then evaluated on the out-of-sample blocks.

As a diagnostic for selection overfitting, we also report the Probability of Backtest Overfitting (PBO), computed using Combinatorial Purged Cross-Validation (López de Prado, 2018; Bailey et al., 2017). PBO estimates how often the strategy chosen as best in-sample falls below the median out-of-sample across train/test splits. This is useful, but not sufficient on its own: as the results show, low PBO can be misleading when the searched strategies are near-duplicates.

The primary performance metric is out-of-sample Sharpe. We also report maximum drawdown and the capacity threshold A^* as secondary diagnostics. When multiple independent runs are available, we report means and standard deviations across runs. For the small replicated comparison in this study, we use a Welch t-test as a descriptive check; a larger confirmatory study would require a multiplicity-robust Sharpe comparison such as Ledoit and Wolf (2008) with stationary bootstrap inference (Politis and Romano, 1994).

5.7 Ablation Design

The ablation study asks two questions. First, does the search architecture itself matter: does MCTS with memory do better than flat LLM generation? Second, does optimizing the robustness score $R(s)$ lead to better out-of-sample strategies than optimizing raw in-sample Sharpe?

We test these questions with a 2×3 design that crosses the objective with the search architecture:

	Objective: $R(s)$	**Objective: Sharpe**
Full	Full-DSR	Full-Sharpe
MCTS-NoPool	NoPool-DSR	NoPool-Sharpe
LLM-Only	LLM-DSR	LLM-Sharpe

Full uses MCTS, the agent pipeline, and the lesson pool. MCTS-NoPool keeps the tree search but removes the lesson pool and novelty gate. LLM-Only removes the tree and generates each proposal from a flat history. The $R(s)$ conditions use the robustness score for node selection, action targeting, and lesson extraction; the Sharpe conditions replace it throughout with raw in-sample Sharpe.

The main comparison is Full-DSR versus Full-Sharpe. This directly tests whether the robustness-aware objective improves the search. The within-column comparisons test whether the tree structure and lesson pool add value beyond ordinary LLM generation.

The fully powered design would require $B = 100$ nodes and $M = 10$ independent runs per cell, or roughly 6,000 backtests, which was beyond our compute and API budget. We therefore treat the once-per-cell grid as directional rather than conclusive. To avoid relying on a single noisy grid, we also evaluate the objective question in two additional ways: offline, by treating each objective as a ranking rule over the pooled strategy corpus in the objective horse-race (Section 7.2), and in-search, by repeating the main Full-DSR/Full-Sharpe comparison four times (Section 7.4).

6 Agentic AI Workflow

Agentic AI tools were used throughout the project, not only as the object of study. This includes helping write this section of the report! The most important uses were literature triage, idea generation, code generation, debugging, experiment design, execution planning, result interpretation, visualization, and report writing. The human contribution was not replaced by the tools: the original research idea, the decision to focus on robustness rather than raw returns, the choice of which negative results were worth pursuing, substantial parts of the literature review and objectives, and the final scientific judgment about where to keep pushing were human-directed. AI assistance was also used in drafting and editing the write-up, including most of the wording, final compression, and clarification pass. Again, the final report uses AI-assisted prose throughout, but the claims and interpretations remain our responsibility.

Project stage	**How agentic AI was used**
Literature and framing	Summarized candidate papers, compared adjacent ideas, and helped turn the broad goal into a testable robustness-vs.-Sharpe question. Human review decided which papers and claims were actually used.
Method design	Helped specify the modular strategy interface, the MCTS actions, the lesson-pool structure, and the ablation grid. Human direction set the core hypothesis and decided that negative results should remain central.
Coding and debugging	Generated and revised Python modules, wrote tests, traced evaluator bugs, diagnosed no-op strategies, and helped identify root-scoring, window-overlap, tune-serialization, and CPCV implementation issues. Human oversight chose which fixes were valid.
Experiment execution	Helped plan budgeted runs, compare saved result files, organize the objective horse-race, and summarize the replicated head-to-head. Human intervention was required to decide which runs were trustworthy and which were only diagnostic.
Visualization and writing	Helped draft tables, captions, the final narrative, and copy-edits. The final report uses AI-assisted prose throughout and particularly in the literature review, but we are responsible for the claims and overall interpretation.

Table 2: Where agentic AI entered the project workflow outside the trading-search system itself.

Overall, the largest acceleration came from iteration speed. Without agentic assistance, building the full search harness, debugging generated strategies, re-scoring saved nodes under corrected protocols, and writing the report in a comprehensive, smooth manner would have been substantially slower. The failures are equally important: AI-generated code could compile while changing no positions, single-run summaries could overstate results, and agentic drafting without a doubt tended to make the framework sound more successful than the evidence supported at times. This is why the final report emphasizes human-audited diagnostics and a negative result rather than forcing self validating AI to produce a positive result.

7 Results

The results are negative, but useful. The robustness objective $R(s)$ does not produce strategies that hold up better out of sample than raw in-sample Sharpe. The main reason is not a single bad term in the reward. It is that the search usually fails before the reward has much to rank: many evaluated nodes behave like small variations of the same strategy.

We begin with a diagnostic study of saved development runs. These runs reveal the central bottleneck—low search diversity—and motivate the safeguards used in later experiments. We then extend the best discovered strategies to the full 1990–2024 panel, test each candidate objective as an out-of-sample ranking rule over the pooled strategy corpus, and finally compare $R(s)$ against raw Sharpe inside the search itself.

7.1 Diagnostic Study: Search Diversity and Failure Modes

The first diagnostic re-scores four saved development searches under the corrected protocol: one larger momentum search and three short budget-capped searches from the moving-average, overnight, and low-volatility seeds. These searches were originally run under an early harness, so all results below come from re-evaluating the saved nodes using the final backtest protocol: 22 annual walk-forward windows over 2000–2021, a held-out 2022–2024 block, and 10 bps one-way transaction costs.

The main diagnostic is K_{eff}, the number of effective return-correlation clusters among the evaluated strategies. If N is large but K_{eff} is small, the search has produced many candidates on paper but only a few genuinely different return streams.

Seed	N	K_{eff}	**PBO**	**Nodes/cluster**
Momentum	89	4	0.029	22.3
Low vol.	11	1	0.043	11.0
MA cross.	13	1	0.200	13.0
Overnight	15	1	0.000	15.0

Table 3: Search-diversity diagnostics. N is the number of evaluated nodes, while K_{eff} is the number of return-correlation clusters at cutoff $\rho^* = 0.7$. The search evaluates many nodes but usually finds only one effective strategy family.

Table 3 shows the basic problem. The three short searches each collapse to a single return-correlation cluster, and even the 89-node momentum search produces only four effective clusters. In other words, MCTS is not really exploring a wide strategy space. It is mostly making local changes around the seed strategy.

This matters for the Deflated Sharpe Ratio. If we used the raw node count N, we would act as though each evaluated node were an independent strategy trial. But many nodes are near-duplicates, so the relevant trial count is closer to K_{eff} than to N. This is why later tables report K_{eff} alongside PBO and use $N_{\text{eff}} = \max(K_{\text{eff}}, 2)$ in the DSR calculation.

The diagnostic runs also exposed two practical failure modes. First, some LLM-generated "improvements" changed the code without changing the portfolio. In the low-volatility tree, one high-scoring child had different module source but produced identical positions to the root across the full sample. We therefore added a position-fingerprint check that rejects children whose realized holdings match their parent.

Second, low PBO can be misleading when search diversity is low. PBO measures whether the in-sample-best strategy falls below the out-of-sample median among the candidates tested. If all candidates are near-clones, the selected strategy can look typical out of sample even when the entire cluster fails in a new regime. The momentum run illustrates this problem: despite a very low PBO, its best discovered strategy lost most of its held-out Sharpe. For this reason, we treat PBO as informative only when reported together with K_{eff}.

The conclusion from the diagnostic study is simple: the binding constraint is not yet the exact form of the reward. It is search diversity. Until the agent generates more genuinely distinct strategies, changing the objective can only have limited effect.

Search diversity across ablation conditions. To measure whether diversity varies by architecture or objective, we extend the K_{eff} diagnostic to the six ablation conditions. Because the full daily return series are not serialized to disk, we use the 22 per-year validation Sharpes stored in each node as a proxy for return correlations. The correlation cutoff and average-linkage clustering method are unchanged ($\rho^* = 0.7$). This yields a coarser estimate than the daily-based K_{eff} reported in Table 3, so the absolute values are not directly comparable; the relative ordering across conditions is the informative quantity.

Cond	Label	Objective	N	$K_{\text{eff}}^{\dagger}$	Nodes/cluster
A	Full-DSR	$R(s)$	82	13	6.3
B	NoPool-DSR	$R(s)$	66	10	6.6
C	LLM-DSR	$R(s)$	94	2	47.0
D	Full-Sharpe	Sharpe	71	5	14.2
E	NoPool-Sharpe	Sharpe	94	5	18.8
F	LLM-Sharpe	Sharpe	97	3	32.3

Table 4: Search-diversity proxy for each ablation condition. † K_{eff} is estimated from 22 annual validation Sharpes rather than daily returns; absolute values are noisier than Table 3 but relative ordering is informative. The LLM-only conditions (C, F) produce near-zero effective diversity despite evaluating the most nodes; MCTS conditions achieve higher cluster counts, confirming that the tree structure does add diversity even when it fails to find improving strategies. Note N varies due to failed coding attempts by the pipeline.

The most striking finding is in the LLM-only conditions. Condition C evaluates 94 nodes but produces only $K_{\text{eff}} = 2$ effective clusters (47 nodes per cluster); condition F has 97 nodes and $K_{\text{eff}} = 3$. Without the tree to direct proposals toward underexplored regions, the undirected LLM regenerates highly correlated ideas run after run. The MCTS conditions achieve meaningfully higher cluster counts ($K_{\text{eff}} = 10$–13 for A and B), confirming that tree structure does increase proposal diversity. The paradox is that A and B achieve higher diversity than C and F while still showing early lock-in (Table 8): the tree explores a wider strategy space but concentrates its *selection* on the first strong node found, so the best-node identity never changes even as the pool of evaluated strategies grows. Diversity of generation and diversity of selection are different problems, and both need to be solved.

Sensitivity to the correlation cutoff. Table 5 extends the proxy K_{eff} analysis to five correlation cutoffs, $\rho^* \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, using the same per-year Sharpe proxy and average-linkage method as Table 4. The relative ordering across conditions is stable at every cutoff: LLM-only conditions (C, F) show only $K_{\text{eff}} = 2$–3 even at $\rho^* = 0.5$, while the MCTS conditions (A, B) achieve 7–8 clusters at that same loose threshold. Absolute cluster counts grow monotonically with ρ^* for all conditions, as expected. Crucially, the near-zero effective diversity of LLM-only architectures holds at every cutoff tested, confirming it is not an artifact of the default $\rho^* = 0.7$ choice.

Cond	Label	N	$\rho^*=0.5$	$\rho^*=0.6$	$\rho^*=0.7$	$\rho^*=0.8$	$\rho^*=0.9$
A	Full-DSR	84	7	9	10	14	20
B	NoPool-DSR	89	8	9	10	12	18
C	LLM-DSR	94	2	2	2	3	8
D	Full-Sharpe	72	3	4	5	8	13
E	NoPool-Sharpe	94	2	4	5	5	11
F	LLM-Sharpe	97	3	3	3	5	11

Table 5: $K_{\text{eff}}^{\dagger}$ proxy across five correlation cutoffs (ρ^*). The LLM-only conditions (C, F) show near-zero effective diversity at every cutoff; MCTS conditions (A, B) consistently achieve higher cluster counts. The $\rho^* = 0.7$ column reflects current tree state (N as shown); condition A shows $K_{\text{eff}} = 10$ here vs. 13 in Table 4 because budget-extension nodes ($N = 84$ vs. 82 there) merged into existing clusters. † Proxy from 22 per-year Sharpe sequences; see Table 4 caption.

7.2 The Objective as a Selector: An Out-of-Sample Horse-Race

The main question is whether $R(s)$ is a better selector than raw in-sample Sharpe. To test this directly, we treat each objective as a ranking rule over the same set of strategies. We pool all saved strategy nodes, remove

duplicates by position fingerprint, and obtain 153 behaviorally distinct strategies from 174 evaluated nodes. The pool is not balanced across seeds: the 89-node momentum tree from Table 3 alone accounts for roughly half of the evaluated nodes that feed this corpus, so the horse-race result should be read as informative mainly about momentum-family strategies, and generalizing it to the other seeds should be done cautiously. For each strategy, we recompute seven candidate objectives on the in-sample period and ask how well each objective predicts realized Sharpe in two held-out blocks: a forward out-of-sample block (2022–2024) and a backward structural-robustness block (1990–1999). The 1990–1999 block predates the 2000–2021 search window rather than following it, and covers a different microstructure regime (pre-decimalization tick sizes, a different listed universe), so we treat it as a robustness check rather than a genuine out-of-sample test; results there are not directly comparable to the forward block, and the 756-day signal warm-up window means the very earliest evaluations in the 1990s may use a truncated history. A useful objective should rank future winners above future losers.

Objective	**IS ρ**	**Forward OOS 2022–24**		**Backward/structural 1990–99**	
	(control)	ρ [95% CI]	top-5	ρ [95% CI]	top-5
Sharpe (null)	0.905	0.562 [.42, .68]	+0.252	0.661 [.54, .75]	+1.785
DSR	0.997	0.537 [.40, .66]	+0.074	0.673 [.55, .77]	+2.040
$R(s)$	0.961	0.562 [.43, .68]	+0.074	0.673 [.55, .76]	+2.040
Sharpe$\cdot\varphi_{\text{stab}}$	0.924	0.523 [.38, .64]	+0.292	0.635 [.50, .73]	+1.751
Sharpe$\cdot\varphi_{\text{cost}}$	0.817	0.543 [.40, .66]	+0.292	0.647 [.52, .74]	+1.751
DSR$\cdot\varphi_{\text{stab}}$	0.993	0.530 [.39, .65]	+0.074	0.660 [.54, .75]	+2.040
Sharpe$\cdot\varphi_{\text{stab}} \cdot \varphi_{\text{cost}}$	0.833	0.535 [.39, .65]	+0.292	0.636 [.50, .73]	+1.751

Table 6: Candidate objectives as selectors over 153 distinct strategies, evaluated against a forward out-of-sample block (2022–2024) and a backward structural-robustness block (1990–1999, pre-decimalization microstructure; see discussion above). ρ is the Spearman correlation between the objective and realized Sharpe in each block. The top-5 columns report the mean realized Sharpe of the five strategies each objective selects. Corpus mean Sharpe is +0.279 in 2022–2024 and +1.242 in 1990–1999. The confidence intervals shown treat the 153 strategies as independent draws; §7.1 and the discussion below note that the corpus collapses to a much smaller number of effective return clusters, so these intervals likely understate the true uncertainty (see note below).

The table gives a simple answer: no robustness objective clearly beats raw Sharpe. In the forward 2022–2024 block, all rank correlations are close together, and $R(s)$ has the same correlation with realized Sharpe as raw in-sample Sharpe. In the backward 1990–1999 block, the DSR-based objectives have slightly higher point estimates, but the differences are small relative to the uncertainty. The robustness machinery does not produce a meaningfully better ordering of strategies.

A caveat on that uncertainty: the confidence intervals above are computed by treating the 153 strategies as independent draws for the Spearman correlation. But §7.1 shows this corpus collapses to a small number of effective return clusters (K_{eff} of 1–4 per seed-level search), so the realized sample of genuinely distinct return streams behind these 153 labels is much smaller than 153. This is the same correlated-trials problem the rest of the paper is diagnosing, applied to the paper's own headline evidence: the true intervals are almost certainly wider than shown, which makes the "no objective is clearly better" conclusion even safer, but it is currently being read off of intervals that are too narrow.

Pooling across all five saved seed-level trees (151 nodes total), the proxy K_{eff} at $\rho^* = 0.7$ using per-year Sharpe sequences is estimated at $K_{\text{eff}} = 6$ for the 99 nodes with full 22-window histories (momentum and reversal seeds, the most reliable estimate) and $K_{\text{eff}} = 11$ when all 151 nodes are included after truncating to the 5-window common denominator of the shorter-run seeds (likely an overestimate due to noisier correlation estimates). Even the generous upper bound of 11 is far below 153, confirming that the effective sample behind the horse-race is an order of magnitude smaller than the label count. Cross-seed pooling roughly doubles effective diversity relative to the best single seed ($K_{\text{eff}} = 4$ for momentum alone), but 6 clusters across 99 nodes still implies roughly 16 nodes per effective strategy family. The confidence intervals reported in

Table 6 should be read with this in mind; a cluster/block bootstrap over return clusters would widen them substantially, but is left as remaining work (the per-strategy held-out returns needed for that recomputation are not archived).

The top-5 results point to a more important story than the choice of objective. In 2022–2024, the top strategies selected by raw Sharpe already underperform the +0.279 corpus average, and the top strategies selected by every robustness-weighted objective do worse still. No in-sample objective in this table beats blind selection of the corpus average in the hard regime. The headline finding of this block is therefore less about which objective ranks strategies and more about non-stationarity: strategies discovered on 2000–2021 data largely do not transfer to 2022–2024, regardless of how they were selected. By contrast, selection works better in the 1990–1999 structural-robustness block, where the top-5 strategies selected by every objective beat the corpus mean – though, as noted above, that block differs enough in microstructure that it should not be read as evidence the method works "out of sample" in the ordinary sense.

The DSR term deserves a more precise claim than "it does not help." DSR is almost perfectly correlated with in-sample Sharpe in this table (IS $\rho = 0.997$), so as a cross-sectional selector over a fixed pool of candidates it is necessarily close to a monotone transform of Sharpe, and the horse-race shows exactly that. But DSR was not designed as a cross-sectional ranker: its purpose is to deflate strategies discovered later in a sequential search by a growing trial count, a mechanism this horse-race holds fixed and therefore cannot exercise. The well-powered test here (the 153-strategy horse-race) evaluates the stability and cost penalties as selectors, where it is informative – the non-deflated objectives give the best held-out top-5 average, though not by enough to claim a decisive improvement – and suggests a better version of the objective might use DSR as a filter and rank surviving strategies by Sharpe times the stability and cost penalties. The claim "DSR specifically does not help" is supported only by the sequential, in-search comparison ($M = 4$, §7.4), which is underpowered. The broader conclusion that the objective choice cannot matter much while diversity is this low is unaffected either way.

7.3 The Full Ablation Grid

We also ran the full 2×3 ablation grid once per cell on the moving-average seed. The grid crosses the search architecture with the objective used during search. Because each cell is only one run, this table is descriptive rather than conclusive.

	Objective: $R(s)$	**Objective: Sharpe**
FULL (MCTS + lessons)	A: +0.387 (PBO 0.03)	D: +0.199 (PBO 0.06)
MCTS-NOPOOL	B: +0.474 (PBO 0.00)	E: +0.536 (PBO 0.37)
LLM-ONLY	C: +0.407 (PBO 0.10)	F: +0.331 (PBO 0.09)

Table 7: Single-run 2×3 ablation grid on the moving-average seed at a budget of approximately $N = 30$ evaluations per cell. Each cell reports held-out 2022–2024 Sharpe and PBO for that condition's best discovered strategy.

Read alone, the grid seems to favor $R(s)$: it beats raw Sharpe in two of the three architecture rows. But this is exactly the kind of conclusion a single LLM-agent run cannot support. The diagnostic study showed that each search explores only a small number of effective strategy clusters, so one run per cell mostly reflects sampling noise. We therefore use the grid only as a directional snapshot. The replicated head-to-head in the next section is the more relevant test of whether the apparent $R(s)$ advantage survives repeated runs.

Budget sensitivity and early lock-in. To probe how stable these results are across evaluation budgets, we extended each cell's search tree in two additional phases, reaching approximately $N = 90$ total evaluations

per condition. Table 8 reports held-out Sharpe at $N \approx 30$ and $N \approx 90$ alongside whether the identity of the best-scoring node changed between budget levels.

Cond	Label	Sharpe @ $N \approx 30$	Sharpe @ $N \approx 90$	Best node stable?
A	Full-DSR	+0.387	+0.387	Yes
B	NoPool-DSR	+0.474	+0.474	Yes
C	LLM-DSR	+0.407	+0.407	Yes
D	Full-Sharpe	+0.199	+0.374	No
E	NoPool-Sharpe	+0.536	+0.430	No
F	LLM-Sharpe	+0.331	+0.337	No

Table 8: Budget sensitivity for the ablation grid. Held-out 2022–2024 Sharpe at two budget levels. All three $R(s)$ conditions returned the same best node at both budgets; all three Sharpe conditions continued to discover better nodes with additional evaluations. Runs within each cell are cumulative (the same search tree is extended), not independent reruns.

The pattern is sharp: every $R(s)$ condition (A, B, C) locked onto its best node within the first ∼30 evaluations and never improved, while every Sharpe condition (D, E, F) continued to find better nodes at higher budgets. In the most striking case, condition D improved from +0.199 to +0.374, nearly closing the apparent gap with A. This early lock-in is consistent with the UCT exploitation pressure being amplified by the DSR penalty: once a node achieves a high $R(s)$ score early in the run, the tree concentrates visits there and explores little else. The consequence is that the apparent advantage of $R(s)$ in the single-run grid at $N \approx 30$ largely reflects which condition was luckier early, not a systematic benefit of the robustness objective.

The lesson pool as a liability. A secondary finding visible in both Table 7 and Table 8 is that removing the lesson pool consistently improves performance. Condition B (NoPool-DSR) outperforms A (Full-DSR) by +0.087 held-out Sharpe at $N \approx 30$, and the gap is maintained at +0.087 at $N \approx 90$—the same best node, found earlier in the no-pool condition. The lesson pool was designed to give the search memory, steering it toward previously successful mechanisms. The evidence here suggests it instead narrows the proposal distribution, causing the search to revisit similar ideas rather than explore genuinely new territory. Ablation B also produces zero PBO versus A's PBO of 0.03, consistent with more predictable OOS behavior when proposals are less constrained by prior lessons. The B-over-A advantage is reproduced in an independent replication (Table 9), where fresh runs from new trees yield +0.527 for B versus +0.367 for A—a +0.160 gap, larger than the original +0.087 and in the same direction.

Independent replication (ma_cross seed, run 2). To verify whether the A-vs-B finding is reproducible, we ran fully independent fresh searches for both conditions A and B from the same moving-average seed, with new trees and new lesson pools (run_idx = 1). Results are shown below alongside the original runs.

Cond	Label	Run 0 (original)	Run 1 (fresh)	B−A gap
A	Full-DSR	+0.387	+0.367	
B	NoPool-DSR	+0.474	+0.527	
	B−A	+0.087	+0.160	

Table 9: Held-out 2022–2024 Sharpe for conditions A and B across two independent runs on the moving-average seed. Run 0 is the original single ablation-grid run; Run 1 uses a new tree and new lesson pool (run_idx= 1).

The B-over-A advantage is reproduced in the fresh run with an even larger gap (+0.160 vs +0.087), confirming that removing the lesson pool consistently helps on this seed. Condition A (run 1) reached $K_{\text{eff}} = 7$ effective

clusters (vs. 13 in run 0) with PBO 0.086 (vs. 0.029), consistent with the stochastic nature of LLM-guided search where the set of discovered strategies varies run to run. Condition B (run 1) achieved $K_{\text{eff}} = 3$ with PBO 0.00, confirming the cleaner OOS behavior that characterizes the no-pool condition.

7.4 Replicated Objective Check

Because the ablation grid uses only one run per cell, we repeated the central comparison: FULL-DSR versus FULL-SHARPE. Each condition was run $M = 4$ times at $B = 50$, after fixing a root-scoring bug that had scored the root node by $R(s)$ even in the Sharpe condition.

Condition	**Objective**	**Single run**	**Replicated, $M = 4$ (mean $\pm$ sd)**
FULL-DSR	$R(s)$	$+0.387$	$+0.189 \pm 0.136$
FULL-SHARPE	Sharpe	$+0.199$	$+0.443 \pm 0.132$

Table 10: Held-out 2022–2024 Sharpe for the central objective comparison. The single run favored $R(s)$, but the replicated comparison reverses the ordering. With only four runs, the difference is not decisive (Welch $t \approx -2.68$, df ≈ 6).

The replicated check changes how the grid should be read. The apparent advantage of $R(s)$ does not survive repeated runs: raw Sharpe has the higher average held-out Sharpe, while two of the four FULL-DSR runs return the unmodified seed. The early lock-in pattern in Table 8 offers a mechanistic explanation: when a FULL-DSR run fails to find a strong node early, the DSR penalty's exploitation pressure causes the tree to collapse back toward the seed, resulting in zero improvement. FULL-SHARPE runs, by contrast, continue exploring at higher budgets, which smooths out the variance across replications. This is not enough to claim that raw Sharpe is definitively better, but it is enough to reject the stronger claim that optimizing $R(s)$ clearly improves out-of-sample performance. Together with the objective horse-race, this points back to the same conclusion: the main bottleneck is search diversity, not the exact form of the reward.

7.5 Cross-Seed Generalization: Overnight-Loser Seed

The moving-average seed provides one data point for the A-vs-D comparison. To test whether the pattern generalizes to a different strategy family, we ran conditions A (FULL-DSR), B (NOPOOL-DSR), and D (FULL-SHARPE) from a second seed: an overnight-loser strategy that ranks stocks by intraday loss (close below open) and holds the worst performers for a daily reversal. This seed has a walk-forward Sharpe of $+0.11$ but regime-stability penalty $\varphi_{\text{stab}} = 0$, giving $R(s) = 0$—an unusually difficult starting point for the DSR conditions.

Cond	**Label**	**Val score**	**Test Sharpe**	**PBO**
A	FULL-DSR	$+0.090$ ($R(s)$)	$+0.537$	0.00
B	NOPOOL-DSR	$+0.147$ ($R(s)$)	$+0.270$	0.01
D	FULL-SHARPE	$+1.071$ (Sharpe)	$+0.274$	0.00

Table 11: Cross-seed results on the overnight-loser seed ($N = 30$ evaluations per condition). Val score is the search-time objective each condition maximized; test Sharpe is held-out 2022–2024 Sharpe of the best discovered strategy.

The overnight-loser seed produces the sharpest evidence in favor of the DSR objective to date. Condition D (FULL-SHARPE) achieved a validation Sharpe of $+1.071$—the highest in-sample score across all conditions in this work—yet its best strategy generalized to only $+0.274$ out of sample, a 74% in-to-OOS drop. Condition

A (FULL-DSR) locked onto its best node at iteration 1 (a plain medium-term momentum signal, $R(s) = +0.090$) and never updated it; nevertheless it delivered a test Sharpe of +0.537, or 96% higher than condition D. This is the clearest single illustration of the intended mechanism: the DSR penalty depressed the apparent attractiveness of condition D's high-Sharpe, high-turnover candidates, while accepting the momentum signal whose lower but more consistent year-to-year Sharpe earned a higher $R(s)$.

Condition B (NOPOOL-DSR) underperforms condition A on this seed (+0.270 vs +0.537), reversing the direction seen on the moving-average seed. The overnight-loser root has $R(s) = 0$, which means the root lesson entered into condition A's pool carried no exploitable information; on the first iteration A found a medium-term momentum signal that immediately earned $R(s) = +0.090$, and this positive lesson then guided subsequent proposals toward related signals. Without the pool, condition B had to rediscover that branch independently and spent its remaining budget on a wider but less productive search. This suggests the lesson pool's effect—helpful or harmful—depends on the quality of the root lesson, which in turn depends on the seed strategy.

The cross-seed results therefore qualify the lesson-pool finding from the moving-average seed: removing the pool is not uniformly beneficial. It helps when the root lesson is already a near-zero-quality signal that risks narrowing proposals toward a poor hypothesis; it hurts when the root lesson happens to point toward a genuinely strong branch.

7.6 Limitations and Concrete Next Steps

The main limitation is scale. The system uses several LLM calls for many proposed strategies, and those calls were costly enough that we could not iterate as freely as we would have liked before running the more serious tests. Early runs looked promising, but after cleaning the protocol and running larger comparisons, the result became more nuanced and mostly negative: the robustness objective did not clearly beat raw in-sample Sharpe, and the search often produced too little effective diversity for the objective to matter much.

This does not make the design unreasonable. The MCTS search and Deflated Sharpe objective were natural choices for the problem: MCTS allocates a limited backtest budget across candidate strategy edits, while DSR directly addresses the multiple-testing problem created by automated strategy search. It is also not obvious that any simple scalar objective should reliably predict out-of-sample Sharpe better than in-sample Sharpe itself, especially when the candidate strategies are highly correlated. The evidence here is therefore not that the approach was misguided, but that at our budget and diversity level its benefits did not appear out of sample.

The most important next step is to fix search diversity before retesting the objective. Concrete changes include increasing the weight on unconstrained DRAFT actions near the root, rejecting proposals whose returns are too highly correlated with an existing cluster, and running several shallow trees from different seeds before combining modules across them. Each change can be evaluated directly by K_{eff} at a fixed node budget. A related diagnostic—computing K_{eff} for each ablation condition rather than only for the development runs—would directly test whether removing the lesson pool or switching to an undirected LLM increases search diversity, since the budget-sensitivity results suggest both may help.

A second priority is to address the early lock-in problem directly. Since the DSR penalty appears to amplify UCT's exploitation bias, one candidate fix is a diversity-aware search objective that explicitly rewards exploring uncorrelated branches, perhaps by penalizing the UCT score of nodes whose return cluster is already well-represented in the tree. Another is to reduce the UCT exploitation constant c adaptively as K_{eff} grows.

The pooled ablation corpus (530 nodes across all six conditions) allows one objective check without new

LLM runs: apply DSR as an admission gate, then rank surviving strategies by

$$\text{Sharpe} \cdot \varphi_{\text{stab}} \cdot \varphi_{\text{cost}}.$$

Table 12 shows survival counts and the top-1 hybrid score at three gate levels. At DSR > 0.5, roughly half the corpus (278/530) survives, but the top hybrid selection is unchanged: the highest-scoring strategy ($\widehat{SR} = 1.23$, $\varphi_{\text{cost}} = 0.95$) already has DSR $= 0.65$, so it clears a moderate gate. At DSR > 0.7 this strategy is excluded and the top-1 hybrid score falls from 0.345 to 0.308. The gate does not robustly improve selection here because the best-by-hybrid strategy achieves its score through high capacity efficiency rather than high deflated Sharpe. Note also that the Spearman $\rho(\text{DSR}, \text{Sharpe}) = 0.64$ across the 530 ablation nodes — lower than the horse-race's 0.997 across the 153-strategy corpus (Table 6) — because the ablation corpus contains many negative-Sharpe nodes where DSR collapses to zero and therefore cannot distinguish among them.

Gate	**Survive**	**%**	**Top-1 hybrid ($\widehat{SR} \cdot \varphi_{\text{stab}} \cdot \varphi_{\text{cost}}$)**
None	530	100%	0.345
DSR > 0.3	351	66%	0.345 (unchanged)
DSR > 0.5	278	52%	0.345 (unchanged)
DSR > 0.7	150	28%	0.308 (changed)

Table 12: DSR-gate hybrid objective on the pooled ablation corpus (530 nodes). Survival counts and top hybrid score at each gate threshold. The top strategy by the hybrid objective has DSR $= 0.65$, so it passes moderate gates but is cut by a strict threshold. A strict gate can exclude the best-by-hybrid strategy, yielding a worse selection.

A larger replicated comparison, such as $M \geq 10$ runs per condition, would be useful only after search diversity is improved.

7.7 Reproducibility and Project Status

The code and saved artifacts are maintained in the shared GitHub repository

https://github.com/RM1505/math285j,

with the active project work on the branch

`feat/baseline-seeds-budget`.

The repository is a large research codebase rather than a polished software package. Because AI assistance was used heavily during development, it contains multiple versions of modules, experiments, saved runs, and analysis scripts. The core pieces needed to understand the results are therefore explained in this report rather than left only to the code.

The main reproducibility goal is that each reported table should trace back to saved code and artifacts rather than manual transcription alone. Relevant files include the backtest and search implementation, analysis scripts such as `objective_horserace.py`, `eval_full_panel.py`, `overnight_run.py`, and `gen_forest.py`, and saved result files such as `horserace_results.json`, `rescore_results.json`, and the replicated head-to-head outputs. The ablation results discussed in Sections 7.3 and 7.4 are stored in `ablation_results.jsonl`.

At submission, the completed pieces are the cleaned search protocol, diagnostic re-scoring, the full-panel extension, the 153-strategy objective horse-race, the once-per-cell ablation grid, the $M = 4$ replicated FULL-DSR/FULL-SHARPE comparison, the budget-sensitivity analysis of Section 7.3, K_{eff} for each ablation condition (Table 4), an independent fresh replication of condition A, cross-seed generalization results on the overnight-loser seed (Section 7.5), the $\text{Var}(\hat{\sigma})$ fallback specification for $K_{\text{eff}} = 1$ (Section 5.3), and the one-

way transaction-cost clarification ($\kappa = 10$ bps one-way; Section 5.1), the ρ^* sensitivity analysis (Table 5), the DSR-gate hybrid objective check (Table 12), and the pooled K_{eff} proxy estimate for the 153-strategy corpus (§7.2). Remaining work includes recomputing the horse-race confidence intervals with a cluster/block bootstrap over return clusters (requires re-archiving per-strategy held-out returns, not currently saved), reporting the four raw per-run held-out Sharpe values behind Table 10 and flagging the seed-collapse runs, improving search diversity, running larger replicated grids, adding richer seeded modules, and replacing the simplified capacity model with a more realistic per-strategy capacity analysis. The first two items require no new LLM runs, only re-analysis of saved results.

8 Conclusion

The central question of this project was whether a robustness-aware search objective could produce strategies that hold up better out of sample than strategies selected by raw in-sample Sharpe. The answer is seed-dependent and nuanced. On the moving-average seed, the robustness objective did not clearly outperform raw Sharpe in either the pooled horse-race or the replicated head-to-head. On the overnight-loser seed, the picture reverses: condition D (Full-Sharpe) achieved the highest in-sample Sharpe of any condition in this work (+1.071) yet generalized to only +0.274 out of sample, while condition A (Full-DSR) accepted a much lower in-sample score but delivered +0.537 held-out Sharpe—a 96% relative gain. This is consistent with the intended mechanism, though a single seed is not a conclusive comparison.

The negative moving-average result is still useful because it identifies where the framework failed more broadly. There are two distinct headline findings, not one. First, search diversity was the main bottleneck: the agent evaluated many nodes, but most behaved like small variations of the same strategy family. In that setting, changing the reward function cannot do much, because the search has not generated enough genuinely different candidates to rank. Second, and separately, the 2022–2024 horse-race shows that no in-sample objective – robustness-weighted or not – beats blind selection of the corpus average in that regime; in-sample winners from 2000–2021 simply do not transfer well into the harder period, which is a statement about non-stationarity dominating objective choice, not a statement about which objective is best. Budget-sensitivity experiments add a further diagnosis: the $R(s)$ objective appears to induce early lock-in, concentrating the search tree around the first strong node found and suppressing later exploration, while raw Sharpe conditions continue discovering better nodes at higher budgets. The lesson pool, intended as a source of memory, appears to narrow the proposal distribution and further reduce diversity. We note one scoping caveat on the DSR term specifically: the well-powered 153-strategy horse-race tests DSR only as a cross-sectional selector, where it is close to a monotone transform of Sharpe by construction and therefore uninformative about DSR's intended sequential-deflation role; the claim that DSR's intended mechanism does not help is supported mainly by the underpowered $M = 4$ in-search comparison.

The project also exposes practical failure modes in LLM-guided research systems. Generated code can compile while leaving portfolio behavior unchanged. Low PBO can look reassuring when the candidate strategies are near-duplicates. Single-run ablations can tell a clean story that reverses under replication. These problems are not specific to this project; they are general risks in agentic search over expensive, overfit-prone evaluations.

The contribution is again diagnostic rather than promotional. We do not show that the proposed robustness objective solves strategy overfitting. Instead, we show why this plausible objective failed in our setting and provide checks—K_{eff}, position fingerprints, joint PBO/diversity reporting, objective horse-races, and budget-sensitivity node tracking—that make those failures visible.

References

Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271, 1999. doi: 10.1016/S0304-405X(98)00052-X.

David H. Bailey and Marcos López de Prado. The Sharpe ratio efficient frontier. *Journal of Risk*, 15(2): 3–44, 2012.

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. doi: 10.3905/jpm.2014.40.5.094.

David H. Bailey, Jonathan Borwein, Marcos López de Prado, and Qiji Jim Zhu. Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, 61(5):458–471, 2014.

David H. Bailey, Jonathan Borwein, Marcos López de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *Journal of Computational Finance*, 20(4):39–69, 2017. doi: 10.21314/JCF.2016.322.

Malcolm Baker, Brendan Bradley, and Jeffrey Wurgler. Benchmarks as limits to arbitrage: Understanding the low-volatility anomaly. *Financial Analysts Journal*, 67(1):40–54, 2011. doi: 10.2469/faj.v67.n1.4.

Jennifer Bender and Taie Wang. Can the whole be more than the sum of its parts? Bottom-up versus top-down multifactor portfolio construction. *Journal of Portfolio Management*, 42(5):39–50, 2016. doi: 10.3905/jpm.2016.42.5.039.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

Riccardo Brutti, Maciej Marowka, and Mihai Cucuringu. Robust orthogonal clustering and applications to equity markets. SSRN Working Paper 5218696, 2025. URL https://ssrn.com/abstract=5218696.

Álvaro Cartea, Mihai Cucuringu, and Qi Jin. Correlation matrix clustering for statistical arbitrage portfolios. SSRN Working Paper 4560455, 2023a. URL https://ssrn.com/abstract=4560455.

Álvaro Cartea, Mihai Cucuringu, and Qi Jin. Detecting lead-lag relationships in stock returns and portfolio strategies. SSRN Working Paper 4599565, 2023b. URL https://ssrn.com/abstract=4599565.

Álvaro Cartea, Mihai Cucuringu, Qi Jin, and Jiahao Zhu. Bottom-up capacity constraints and the limits of anomaly profitability. SSRN Working Paper 5797502, 2025a. URL https://ssrn.com/abstract=5797502.

Álvaro Cartea, Mihai Cucuringu, and Dongwoo Kim. Learning firm characteristics for asset management. SSRN Working Paper 5853103, 2025b. URL https://ssrn.com/abstract=5853103.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. MARS: Modular agent with reflective search for automated AI research. *arXiv preprint arXiv:2602.02660*, 2026.

Stefano Ciliberti and Stanislao Gualdi. Portfolio construction matters. *arXiv preprint arXiv:1810.08384*, 2018. URL https://arxiv.org/abs/1810.08384.

Andrea Frazzini and Lasse Heje Pedersen. Betting against beta. *Journal of Financial Economics*, 111(1): 1–25, 2014. doi: 10.1016/j.jfineco.2013.10.005.

Campbell R. Harvey, Yan Liu, and Caroline Zhu. . . . and the cross-section of expected returns. *Review of Financial Studies*, 29(1):5–68, 2016. doi: 10.1093/rfs/hhv059.

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. doi: 10.1111/j.1540-6261.1993.tb04702.x.

Nour Khelifa, Jules Allier, and Mihai Cucuringu. Cluster-driven hierarchical representation of large asset universes for optimal portfolio construction. In *Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF)*, 2024. doi: 10.1145/3677052.3698676.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006. doi: 10.1007/11871842_29.

Olivier Ledoit and Michael Wolf. Robust performance hypothesis testing with the Sharpe ratio. *Journal of Empirical Finance*, 15(5):850–859, 2008. doi: 10.1016/j.jempfin.2008.03.002.

Weixian Waylon Li, Hyeonjun Kim, Mihai Cucuringu, and Tiejun Ma. Can llm-based financial investing strategies outperform the market in long run?, 2026. URL https://arxiv.org/abs/2505.07078.

Marcos López de Prado. *Advances in Financial Machine Learning*. Wiley, 2018.

Dong Lou, Christopher Polk, and Spyros Skouras. A tug of war: Overnight versus intraday expected returns. *Journal of Financial Economics*, 134(1):192–213, 2019. doi: 10.1016/j.jfineco.2019.03.011.

Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012. doi: 10.1016/j.jfineco.2011.11.003.

Dimitris N. Politis and Joseph P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994. doi: 10.1080/01621459.1994.10476870.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tyler Shumway. The delisting bias in CRSP data. *The Journal of Finance*, 52(1):327–340, 1997.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160, 2024.