Alpha Evolve: Evolving Cross-Sectional Trading Signals and Evaluating Portfolio Backtests

Alex Bethune

January 28, 2026

Abstract

This paper documents the procedures and findings of *Alpha Evolve*, a project that automatically generates candidate trading signals ("alphas") via an evolutionary search process and evaluates them using a portfolio backtesting engine. We describe the data pipeline, signal generation mechanism, evaluation protocol, and risk/portfolio constraints. We then report results from a representative end-to-end run (dated January 18, 2026) including performance metrics for the top-ranked alpha signals and an ensemble selection.

1 Introduction

Designing quantitative trading signals typically involves iterative feature engineering and repeated backtests. Alpha Evolve explores an alternative workflow: define a library of allowable operations and an evaluation objective, then use an evolutionary process to search the space of candidate programs that compute trading signals from market data. The system produces a ranked set of signals and associated backtest artifacts that support analysis and reproducibility.

This project was inspired by the prior Alpha Evolve paper included with this repository [**?**].

2 Project Overview

Alpha Evolve consists of:

- a data layer that loads and aligns per-symbol OHLC time series,
- a program representation for signal logic (expressed as operator graphs/programs),
- an evolutionary engine that mutates programs and selects candidates based on evaluation metrics, and
- a portfolio backtester that converts signal values into positions and computes risk-adjusted performance.

3 Procedures (Methods)

3.1 Data

For experiments, we used a trimmed S&P 500 dataset consisting of 30 symbols with approximately three years of daily OHLC history starting on 2020-01-01. Each symbol file follows a common schema: `time, open, high, low, close` with `time` stored as UNIX seconds.

3.2 Train/Validation Splits

During evolution, candidates are evaluated on a fixed-length train/validation split to reduce overfitting. In the run reported here, the configuration used 840 train points and 360 validation points.

3.3 Evolutionary Search

Candidates are represented as programs composed from an operator library (e.g., rolling means, ranks, elementwise transforms, cross-sectional transforms, and relation-based operators).

A typical run proceeds as follows:

1. **Initialize population:** Randomly generate a population of candidate programs under structural constraints (maximum operator count and operand limits).
2. **Evaluate:** For each program, compute per-day cross-sectional predictions and evaluate using the configured objective (e.g., a Sharpe-like proxy).
3. **Select:** Choose parents via tournament selection.
4. **Vary:** Apply mutation (and optionally crossover) to produce a new population; preserve elites; inject a fraction of fresh random candidates.
5. **Repeat:** Iterate for a fixed number of generations; retain a hall-of-fame of top programs.

In the representative run (`run_Amelia-Turner_g120_seed101_full_overlap_20260118_163833`), the evolutionary configuration used 120 generations, population size 240, tournament size 10, and a high mutation rate.

3.4 Portfolio Backtesting

Top-ranked programs are then backtested in a portfolio simulation. Key settings in the reported run include:

- holding period: 2 bars,
- evaluation lag: 2 bars,
- volatility targeting enabled (configured target `volatility_target = 0.0111`),
- drawdown control enabled (configured `dd_limit = 0.15`),
- ensemble mode enabled with ensemble size 5.

Signals are scaled using a rank-based approach and optionally winsorized (here, `winsor_p = 0.02`). The backtester produces summary metrics (annualized return and volatility, Sharpe ratio, drawdown, turnover) and time series artifacts.

4 Findings (Results)

We summarize the backtest metrics reported for the top five alphas from the representative run dated January 18, 2026. Table 1 reports the key portfolio metrics.

The best single alpha in this run was `Alpha_01` with Sharpe ≈ 1.40 and annualized return ≈ 0.266. The recorded program for `Alpha_01` was:

```
P[s1_predictions_vector = neg(ma5_t)]
```

Table 1: Top-5 backtested alphas from run run_Amelia-Turner_g120_seed101_full_overlap_20260118_163833.

AlphaID	Sharpe	AnnReturn	AnnVol	MaxDD
Alpha_01	1.402	0.266	0.180	0.278
Alpha_02	1.134	0.193	0.169	0.294
Alpha_03	1.368	0.257	0.179	0.273
Alpha_04	1.368	0.257	0.179	0.273
Alpha_05	1.383	0.258	0.177	0.274

4.1 Ensemble Selection

An ensemble of five members was selected (`Alpha_01, Alpha_02, Alpha_05, Alpha_03, Alpha_04`) under a maximum correlation constraint (configured `max_corr = 0.7`).

4.2 Equity Curve Example

Figure 1 shows the time series produced for `Alpha_01`.

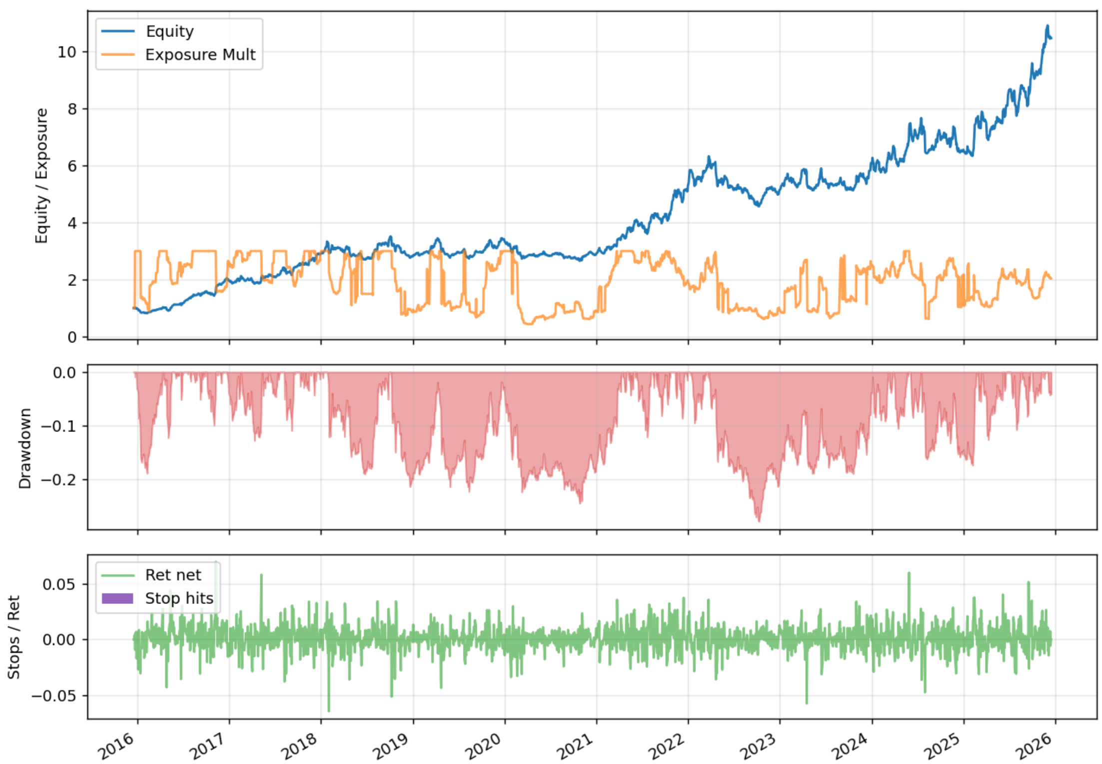

Figure 1: Backtest time series for `Alpha_01` in the representative run.

5 Discussion

The reported run demonstrates that the system can discover simple, low-operator signals (e.g., a negated moving-average feature) that achieve moderate risk-adjusted performance under the given backtest and risk controls. However, stress-test performance in the same summaries indicates sensitivity under adverse scenarios, motivating additional robustness checks and stronger constraints.

6 Limitations

- Results are conditional on the chosen universe, window, and backtest settings.
- Evolutionary search can overfit to the evaluation proxy; stronger cross-validation and out-of-sample testing may be required.
- Many operator-rich programs are difficult to interpret; interpretability constraints or pruning may be needed.

7 Conclusion

Alpha Evolve provides an end-to-end pipeline for generating, ranking, and backtesting cross-sectional trading signals. The pipeline outputs structured artifacts (configs, summaries, and time series) that support reproducible evaluation. Future work should emphasize robustness (e.g., improved stress testing, regime-aware evaluation, and broader universes) alongside performance.

References