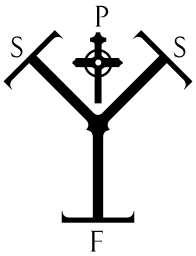

DOCILIS - MITIS - HUMILIS

Tactical Trend Trader: A System and Method for Adaptive Algorithmic Trading via Real-Time Parameter Optimization and Dynamic Reconciliation

Trisagion Developers

support@trisagion.xyz
www.trisagion.xyz

January 29, 2026

Abstract

A system and method for algorithmic trading, hereinafter referred to as the Tactical Trend Trader (3T), is presented, designed to execute trades based on a quantitative, adaptive framework. The system operates as a persistent, stateful process that continuously evaluates market conditions to inform trading decisions. The core operational cyclical control-flow graph comprises four distinct stages: data ingestion, multi-parameter optimization, trade evaluation and execution, and state reconciliation. A primary contribution of this work is a massively parallel optimization engine that tests sufficiently large combinations of parameter permutations in real-time to identify locally optimal configurations. This architecture is a direct response to the documented limitations of conventional predictive models in non-stationary markets, favoring continuous, exploratory adaptation over retrospective prediction. For market state classification, the system utilizes Permutation Entropy to distinguish between predictable, trending regimes and stochastic, non-trending noise. A key innovation is its departure from the classical Kelly Criterion for position sizing; instead, it employs a performance-based heuristic that dynamically modulates a base position size according to recent profitability. Furthermore, risk management is abstracted from individual trades and managed at the portfolio level via a scheduled reconciliation process, obviating the need for conventional stop-loss orders. This synergistic combination provides a robust framework for systematic trading that addresses the fundamental challenge of adaptation lag inherent in history-bound models.

1 Introduction

The domain of algorithmic trading is characterized by a persistent challenge: developing systems that can adapt to non-stationary and often chaotic financial markets [13]. Many conventional systems rely on statically configured parameters, rendering them brittle and susceptible to performance degradation as market dynamics shift. Even sophisticated systems that employ periodic model retraining can fail when confronted with sudden regime shifts, structural breaks, or rare black-swan events. The Tactical Trend Trader (3T) is a computer-implemented

system engineered to address these limitations by integrating real-time adaptability into its core architecture. It is designed to optimize performance by leveraging statistical analysis for market regime detection and a novel set of heuristics for risk and position management. The current architecture of the 3T is the result of a deliberate evolution away from a more conventional, but ultimately less effective, machine learning paradigm. Early system development focused on a regression-based approach using a sophisticated **bootstrap-aggregated ensemble** of heterogeneous learners. This ensemble included a Multi-layer Perceptron (MLP), several gradient-boosted tree models (LightGBM, CatBoost, XGBoost), a Random Forest, and a k-Nearest Neighbours classifier [2,4,5,8] provided by AutoGluon [18]. This predictive model was re-trained hourly on all accumulated historical observations of the unbiased continuous evaluations with the goal of forecasting the expected profit and loss of potential portfolio positions. During inference, a trade would only be executed against live risk if the predictor scored positively. However, over a period of several months of study, this predictive approach consistently underperformed. The core issue identified was that models trained on historical data are fundamentally retrospective. Their learned mappings from market features to expected outcomes become stale in the face of new market dynamics, creating a performance-costly "adaptation lag" that persists even after incorporating changes in the next model retrain. The pivotal insight from this empirical investigation was that randomized, out-of-sample feature values were consistently superior to the carefully trained tabular predictors. This finding supports a central thesis: for highly unpredictable environments, **online stochastic exploration is more effective than classic supervised learning**. Consequently, the machine learning aspect was removed entirely in favor of the current paradigm. The 3T system now operates in a continuous loop, analyzing market data to first identify predictable, trending environments and then deploying capital with a position size dynamically calibrated to recent system performance. This paper details the mathematical framework, system architecture, and unique operational logic of the 3T. The primary contributions of this method are:

1.1 A Real-Time Optimization Engine

A mechanism that continuously stress-tests a large number of parameter permutations against live market data. This engine is the direct, successful replacement for the abandoned machine learning ensemble. This optimization includes both momentum-following ("breakout") [12] and mean-reverting ("swing") [15] entry strategies simultaneously, allowing for greater tactical flexibility.

1.2 A Hybrid Risk Management Model

A novel heuristic for position sizing based on recent performance, coupled with a dynamic, reconciliation-based approach to position control that replaces static, per-trade stop-losses.

2 Related Work

The 3T's architecture is informed by established practices in quantitative finance but represents a novel synthesis that prioritizes real-time adaptation. This section surveys the two competing paradigms—ensemble-based prediction and stochastic optimization—that were empirically tested during the system's development.

2.1 Ensemble Methods in Financial Prediction

Ensemble methods, such as bootstrap aggregating (bagging) [2] and gradient-boosted trees [4,5,8], have become standard tools for modeling tabular financial data. Their ability to reduce variance and capture complex, non-linear interactions makes them attractive for tasks like predicting risk-adjusted returns [17]. To combat the non-stationarity of financial time series, online learning and incremental model updating schemes are often employed [14]. Despite their power, these methods share a fundamental limitation: they rely on a "past-to-present" update rule. A model's knowledge is exclusively derived from historical data. Consequently, they inherit an intrinsic "adaptation lag," rendering them vulnerable to sudden regime shifts where past patterns no longer hold predictive power.

2.2 Stochastic Optimization in Finance

An alternative paradigm is stochastic optimization, which includes methods such as simulated annealing [10], evolutionary strategies [6], and random search [3]. The primary strength of these techniques lies in their ability to explore a broad search space without being confined to a deterministic gradient or patterns learned from historical data. In finance, they are commonly applied to offline tasks, such as portfolio construction or the hyper-parameter tuning of other models.

2.3 The 3T's Position: Online Stochastic Policy Generation

The 3T system's core innovation is to take the principles of stochastic optimization and apply them not as an offline tuning process, but as the central, online, real-time decision-making mechanism. The system's Parameter Optimization Engine functions as a **continuous stochastic policy generator**. This approach directly addresses the adaptation lag of ensemble methods by continuously testing new parameter sets against live market data, making it a more suitable paradigm for highly dynamic and unpredictable markets. The fundamental differences between these two approaches are summarized in Table 1.

3 System Architecture

The 3T system is a modular architecture comprising four main components (see Fig. 1): a Data Ingestion Module, a Parameter Optimization Engine, a Trade Evaluation and Execution Engine, and a State Reconciliation Module. This separation of concerns ensures robustness and scalability.

3.1 Data Ingestion Module

This module serves as the sensory input for the entire system, acquiring market data from two primary sources:

- **Real-time Websocket Stream:** Provides low-latency access to immediate price updates (tick data), crucial for the Parameter Optimization Engine.
- **Historical API Endpoint:** Fetches historical Open/High/Low/Close (OHLC) data, providing aggregated into one-minute volatility bars. This provides the necessary historical context for macro-level analysis like Permutation Entropy calculation.

3.2 Parameter Optimization Engine

This engine operates as a continuous, massively parallel simulation layer that runs concurrently with the live trading system. Its purpose is to find the most profitable set of parameters for the core trading algorithm in the current market environment strictly as a function of out-of-sample performance. This architecture embodies a "shotgun" or global random search methodology, a direct response to the failures of retrospective predictive models.

Instead of iterating from a single base parameter set, the engine spawns thousands of independent simulations, each configured with parameters drawn randomly from predefined, permissible ranges. Each simulation, or 'run', is a completely independent instance that evaluates its unique strategy against the live market data feed. These runs are ephemeral and exist to test a hypothesis; they either succeed by generating a positive profit-and-loss (PnL) or fail and are terminated.

The system's adaptation does not come from mutating a 'base' parameter set in a hill-climbing fashion. Instead, it emerges from the selection process performed by the State Reconciliation Module, which observes the entire population of active runs. By identifying the strategies that are currently profitable, the Reconciliation Module derives the desired portfolio state. This 'clonal selection' model, inspired by the adaptive immune system, allows the system to respond to market changes by favoring the 'antibodies' (parameter sets) that are most effective against the current 'antigen' (market conditions), addressing the adaptation lag of traditional models.

The key parameters under optimization include:

- **Permutation entropy parameters:** The embedding dimension d and time lag τ.
- **Market state thresholds:** The specific value of Permutation Entropy, H_{trend}, below which the market is considered 'trending'.
- **Max duration:** The maximum time, in seconds, that a single simulation ('run') is allowed to process data.
- **Max direction reversal:** The maximum time a run will attempt to find a market direction before exiting.
- **APR target:** The base Annual Percentage Rate (APR) that serves as a benchmark for the fitness function.
- **Rolling APR minutes:** The time window over which the system's APR stability is calculated.
- **Decision distance seconds:** The minimum interval between successive trade evaluations.
- **System type:** The entry logic, supporting "breakout" or "swing" methodologies.

The most profitable parameter sets discovered by this engine are implicitly selected by the Reconciliation Engine, which uses their success to guide the live, risk-on portfolio.

Table 1: Paradigmatic Comparison of Trading System Approaches.

Feature	Tabular Predictor (Ensemble)	Stochastic Engine (3T)
Core Task	Prediction of future Profit & Loss based on learned patterns.	Adaptation of system parameters to maximize immediate profitability.
Data Reliance	Retrospective: Trained exclusively on historical data.	Prospective: Tested continuously on live, real-time data.
Adaptation Mechanism	Periodic Batch Retraining (e.g., hourly) on accumulated data.	Continuous Real-Time Perturbation and evaluation of parameters.
Adaptation Latency	High (minutes to hours). The model is stale between retraining intervals.	Near-Zero. Adaptation occurs at the frequency of the decision epoch.
Primary Failure Mode	Stale mapping during regime shifts, leading to poor decisions based on outdated patterns.	Inefficient exploration if the perturbation variance is poorly tuned.

4 Mathematical Framework and Operational Logic

The core of the 3T is a continuous loop executed by the Trade Evaluation and Execution Engine, which acts upon the optimal parameters supplied by the Parameter Optimization Engine. Each iteration performs a rigorous sequence of data analysis and trade management operations.

4.1 Market Regime Analysis via Permutation Entropy

To prevent trading in stochastic, unpredictable market conditions, the system first performs a market regime analysis. This methodology is chosen for its robustness to noise and its model-free nature [1]. The procedure is detailed in Algorithm 1.

Algorithm 1 Market Regime Classification

Require: Time series $X = \{x_1, x_2, \ldots, x_N\}$
Require: Embedding dimension d, Time lag τ
Require: Threshold H_{trend}
  Construct embedding vectors from X with lag τ
  Calculate relative frequency p_i for each permutation pattern π_i of order d
  $H(d) \leftarrow -\sum_{i=1}^{d!} p_i \log_2 p_i$
  if $H(d) < H_{trend}$ **then**
    return Trending
  else
    return Noisy
  end if

4.2 System Entry Logic: Breakout and Swing Types

Once a trending market is confirmed, the system employs its configured entry logic based on the system_type parameter.

- **Breakout Logic:** This is a momentum-following strategy. A long entry is considered if the current price exceeds the highest high of the last N periods, and a short entry is considered if the price falls below the lowest low of the last N periods.
- **Swing Logic:** This is a mean-reversion strategy. It identifies short-term price extensions away from a central tendency (e.g., a moving average) and seeks to enter at a more favorable price point.

4.3 Performance Epochs and the Portfolio Lifecycle

To manage its continuous, massively parallel simulations, the system groups them into discrete **performance epochs**. These epochs can be thought of as distinct generations of trading strategies, each with a defined beginning and end. This mechanism is fundamental to how the system crystallizes gains and learns from its own history.

At any given time, thousands of independent simulations are active. This entire population represents the **live cohort**, a generation of strategies collectively testing hypotheses against current market conditions. This cohort is unique because it has not yet been finalized or assigned to a historical epoch.

A system-wide monitoring component observes the portfolio's aggregate profit-and-loss. This monitor is detached from the success of any individual simulation; its sole purpose is to determine when the

portfolio *as a whole* has achieved a significant, predefined profit target. When this target is reached, the monitor triggers a state transition: it declares the live cohort a success and finalizes it by assigning it to a new, unique performance epoch.

This action has two critical, simultaneous effects:

1. It creates an immutable historical record of the generation of strategies that produced the successful outcome.
2. It signals the termination of every simulation within that cohort, thus "banking the win" for the entire portfolio and forcing a reset with a new cohort of simulations. This serves as the system's primary take-profit mechanism.

4.4 Position Sizing Heuristic and APR Trend Analysis

When a valid entry signal is generated, the system determines its risk exposure based on recent profitability, measured by its Annual Percentage Rate (APR).

4.4.1 Annual Percentage Rate (APR) Calculation

The primary measure of a run's performance is its APR, calculated as follows:

$$\text{APR} = \frac{(B_{cur} - B_{start})}{D_h} \times 24 \times 365.24 \times \frac{1}{B_{start}} \quad (1)$$

where B_{cur} is the current virtual balance, B_{start} is the starting balance, and D_h is the duration of the run in hours. This metric annualizes the run's return, providing a standardized way to compare the performance of runs with different lifespans.

4.4.2 APR Gating Condition

A core rule of the risk management framework is that position size will **only** be increased if the system's rolling APR is trending positively and is above the dynamically calculated APR target. This ensures that risk is escalated only during periods of sustained, proven profitability.

4.4.3 Performance-Based Sizing Heuristic

If the APR gating condition is met, the system calculates the appropriate position size. It deliberately eschews a direct implementation of the Kelly Criterion, which can be sensitive to estimation errors in its inputs. Instead, it employs a novel performance-based heuristic that modulates a base risk position size (S_{base}) based on the relative performance of current versus historical simulations.

First, the Kelly Percentage ($K\%$) is calculated for a given set of runs. This is derived from two primary metrics:

- **Win Rate (W):** The fraction of runs with a positive final profit-and-loss (PnL).
- **Reward/Risk Ratio (R):** The ratio of the average PnL of winning runs to the average PnL of losing runs.

The Kelly Percentage is then given by:

$$K\% = W - \frac{1 - W}{R} \quad (2)$$

Using this, the system defines two key inputs for the sizing heuristic:

- H_{cur}: The Kelly Percentage calculated from the **live cohort**—the population of currently active simulations not yet assigned to a historical epoch.
- H_{hist}: The aggregate Kelly Percentage derived exclusively from all previously **finalized performance epochs**.

This "positive expectancy gate" on H_{hist} ensures the current strategy is compared against a robust "winner's index" rather than a global average that may be diluted by noise. The performance factor, P_{kelly}, is then defined as the relative change between these two Kelly percentages:

$$P_{\text{kelly}} = \begin{cases} 0, & H_{\text{hist}} = 0 \\ \dfrac{H_{\text{cur}} - H_{\text{hist}}}{|H_{\text{hist}}|}, & \text{otherwise} \end{cases} \quad (3)$$

The factor P_{kelly} is then clipped to a symmetric bound $[-\tau, \tau]$ (default $\tau = 0.5$) and floored at -0.98. The final position size (S_{final}) is calculated by adjusting the base size (S_{base}, a configured fraction of the total portfolio equity) by this performance factor:

$$S_{\text{final}} = S_{\text{base}}\big(1 + \text{clip}(P_{\text{kelly}}, -\tau, \tau)\big) \quad (4)$$

where the clip function is defined as:

$$\text{clip}(x, \min, \max) = \begin{cases} \max & \text{if } x > \max \\ \min & \text{if } x < \min \\ x & \text{otherwise} \end{cases} \quad (5)$$

Derivation & Edge-Case Handling Table 2 summarises how edge cases observed in production are handled.

Algorithmic Pseudocode Algorithm 2 presents the high-level steps for symbol level expectancy regime based position sizing (see Fig. 2).

Algorithm 2 Kelly-adjusted position sizing

$H_{\text{cur}} \leftarrow$ `_calculate_kelly_metrics(..., ''new block height'')`
$H_{\text{hist}} \leftarrow$ `_calculate_kelly_metrics(..., ''existing block heights with positive performance'')`
if H_{cur} is `None` **or** H_{hist} is `None` **then**
  $S_{\text{final}} \leftarrow S_{\text{base}}$
else
  $P \leftarrow \dfrac{H_{\text{cur}} - H_{\text{hist}}}{|H_{\text{hist}}|}$
  $P \leftarrow \text{clip}(P,\ -\tau,\ \tau)$
  $S_{\text{final}} \leftarrow S_{\text{base}} \times (1 + P)$
end if
return S_{final}

4.4.4 Execution and Risk Management

1. **Take-Profit**: A take-profit goal is calculated dynamically based on market conditions. The base `apr_target` is scaled by a "volatility goal," a multiplier derived from the market's recent volatility. This allows the system to set more ambitious profit targets during periods of high volatility and more conservative targets when the market is calm. The calculation is governed by several key parameters: `min_goal`, `max_goal`, `min_goal_weight`, and `max_goal_weight`. A run is exited if its APR exceeds this dynamically adjusted target.

2. **Risk Management**: The 3T explicitly omits traditional, per-trade stop-loss orders, which are susceptible to transient volatility. Instead, risk is managed holistically through State Reconciliation.

5 State Reconciliation Module

The State Reconciliation Module functions as a dynamic, portfolio-level risk manager (see Fig. 3). Its operation is scheduled at a configured interval (e.g., every 10 minutes) rather than being event-driven by price action. This prevents high-frequency, reactive adjustments and allows positions time to mature. A key aspect of this module is its symbiotic relationship with the Parameter Optimization Engine. The engine's continuous simulations provide an intelligent, real-time benchmark of expected performance for thousands of alternative parameter configurations. During each reconciliation cycle, the module assesses the aggregate health of all open positions not against a static rule, but against a filtered dynamic benchmark that excludes historically unprofitable regime states. If the actual portfolio performance deviates significantly from the simulated expectancy of the top-performing virtual configurations, or if the market regime has shifted to 'noisy', the module takes corrective action. This action is based on the simulated success of the best alternative parameter set and may involve reducing or closing a position entirely. This process serves as an intelligent, system-wide stop-loss mechanism. A position is not closed simply because it is losing money, but because the system has discovered, via simulation, that better parameter configurations exist for the current market conditions. To prevent excessive trading fees from minor adjustments ("churn"), the module will only execute a change if the required adjustment in position size is above a minimum financial or percentage threshold. Furthermore, to ensure the reliability of risk-on decisions, multiple observer nodes are utilized to achieve consensus, preventing a runaway process from continuously adjusting a position without checks and balances [11].

6 Configuration Parameters

The reconciliation engine is driven by a small set of runtime configuration keys. Table 3 lists the keys, their defaults, and a brief description.

7 Conclusion

The Tactical Trend Trader (3T) provides a robust and complete framework for adaptive algorithmic trading. Its novelty and efficacy stem from a deliberate architectural choice to favor real-time, exploratory adaptation over retrospective prediction. This design was the direct result of empirical studies where a sophisticated, bootstrap-aggregated machine learning ensemble failed to outperform the stochastic approach in simulated, non-stationary market conditions. The system's strength lies in the synergistic combination of its core components: (1) the use of Permutation Entropy for robust market regime filtering, (2) a massively parallel, real-time Parameter Optimization Engine that ensures the system

Table 2: Edge-case handling for the Kelly-based performance factor.

Edge case	**Handling in code**	**Paper description**
$H_{\text{hist}} = 0$ (no historical data)	Return base size unchanged	"If no historical Kelly can be computed, the engine falls back to the unadjusted base size."
Insufficient sample size < 10 runs	Return `None` $\rightarrow$ fallback	"Kelly is only calculated when at least ten qualifying runs exist; otherwise the metric is unavailable."
$\lvert P_{\text{kelly}} \rvert > \tau$	Clip to $\pm\tau$	"A hard cap prevents runaway position scaling."
$P_{\text{kelly}} < -1.0$	Floor at -0.98	"The floor avoids a sign reversal that would otherwise flip the position direction while maintaining a small amount of pass through sampling."

Table 3: Key configuration parameters used by the reconciliation engine during testing.

Config key	**Default**	**Meaning**
`kelly_threshold`	0.5	Symmetric cap for the performance factor P_{kelly}.
`risk_pos_percentage`	0.0016180339887	Fraction of account equity used as the base risk size.
`minimum_trade_threshold`	20.0 (USD)	Minimum notional size before a trade is emitted.
`position_staleness_timeout`	300 s	Max age of a local position record before it is considered stale.

remains adapted to current market dynamics, (3) a unique performance-based heuristic for dynamic position sizing, and (4) an active reconciliation module that provides intelligent, portfolio-level risk management. By systematically filtering for predictable market environments and embracing continuous optimization at inference time, the 3T system is designed to navigate complex financial markets, addressing the critical "adaptation lag" that limits the performance of models reliant on historical data.

8 Future Work

The earlier implementation of machine learning models did show effectiveness on shorter time frames. Some next steps include investigating hybrid approaches that combine the strengths of both paradigms; a lightweight predictive model could be used in combination with the stochastic engine so they are complementrary inputs to the portfolio. Additionally, there is interest in Hurst processes for complementing entropy thresholds to optimize periods of flat/highly oscillatory conditions. Correlation study between market oscillation and system maximum adverse excursions per profit and loss block.

References

[1] C. Bandt and B. Pompe, *Permutation Entropy: A Natural Complexity Measure for Time Series*, Phys. Rev. Lett. **88**, 174102 (2002).

[2] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[4] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," *arXiv preprint arXiv:1706.09516*, 2017.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[6] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[7] M. Lopez de Prado, *Advances in Financial Machine Learning*, Wiley, 2018.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[9] J. L. Kelly, Jr., *A New Interpretation of Information Rate*, Bell System Technical Journal, **35**(4), 917-926 (1956).

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] L. Lamport, R. Shostak, and M. Pease, *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, **4**(3), 382-401 (1982).

[12] E. Lefèvre, *Reminiscences of a Stock Operator*, George H. Doran Company (1923).

[13] R. S. Tsay, *Analysis of Financial Time Series*, Wiley, 2005.

[14] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249-289, 2021.

[15] G. D. Taylor, *The Taylor Trading Technique*, self-published (1950).

[16] Van Tharp Institute, *Peak Performance 101 Course for Traders and Investors*. Available online: https://vantharpinstitute.com/course/peak-performance-course-for-traders-and-investors/. (Accessed on: September 8, 2025).

[17] B. M. Henrique, V. A. Sobreiro, and H. Kimura, "Literature review: Machine learning techniques applied to financial market prediction," *Expert Systems with Applications*, vol. 124, pp. 226-251, 2019.

[18] Erickson, Nick and Mueller, Jonas and Shirkov, Alexander and Zhang, Hang and Larroy, Pedro and Li, Mu and Smola, Alexander, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data" *arXiv preprint arXiv:2003.06505, 2020*

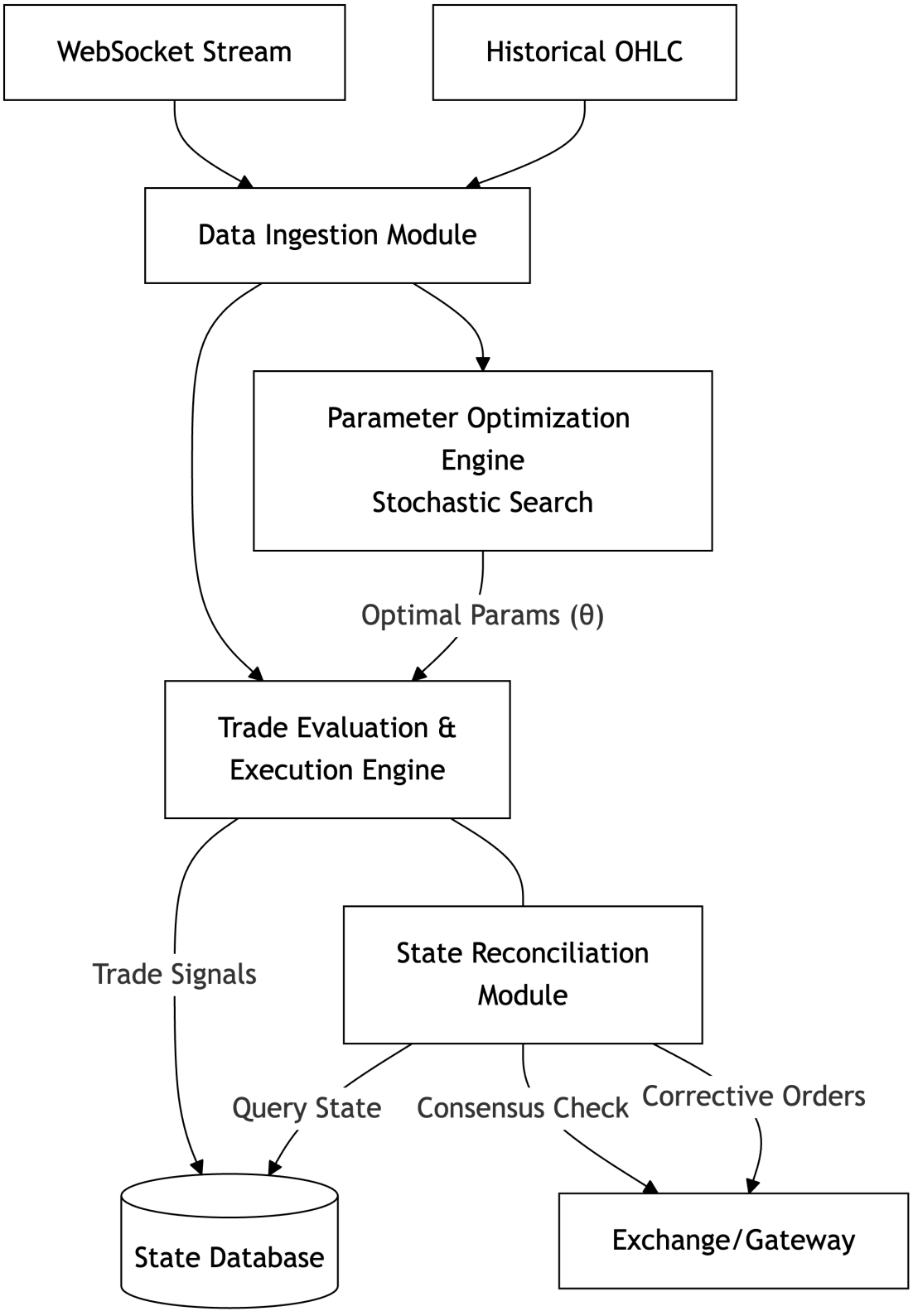

Figure 1: System Architecture

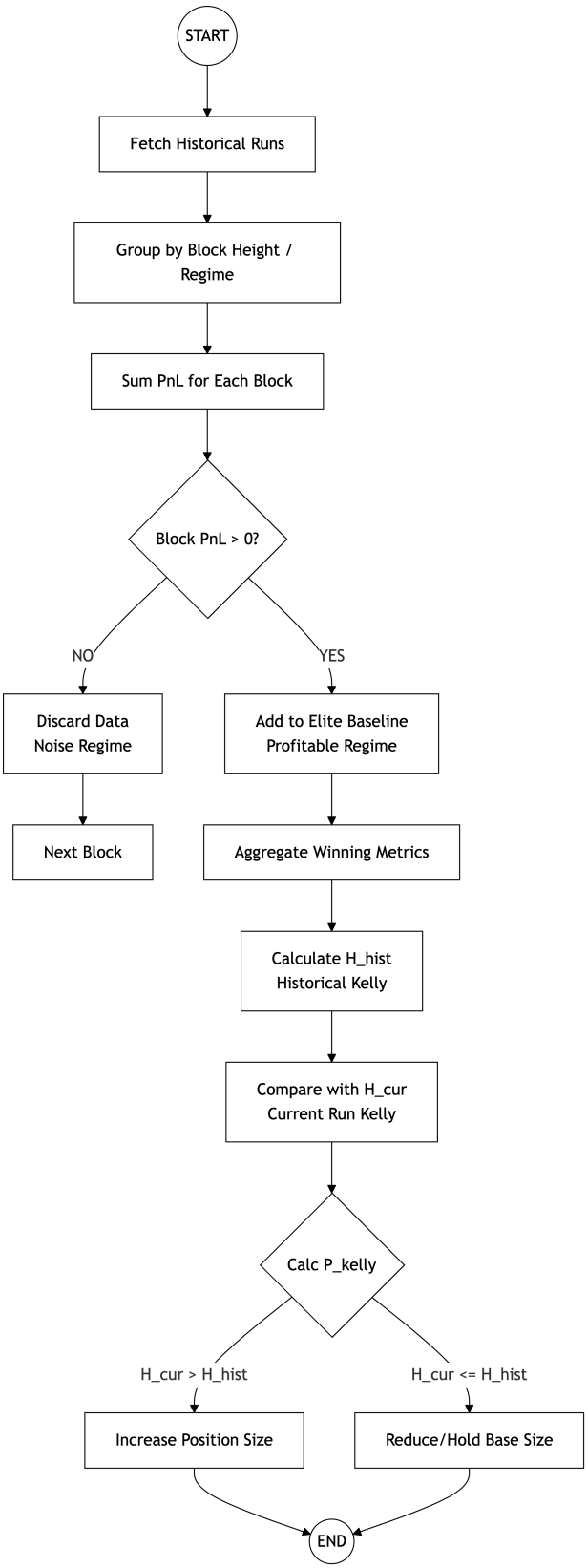

Figure 2: Expectancy Regime Filter

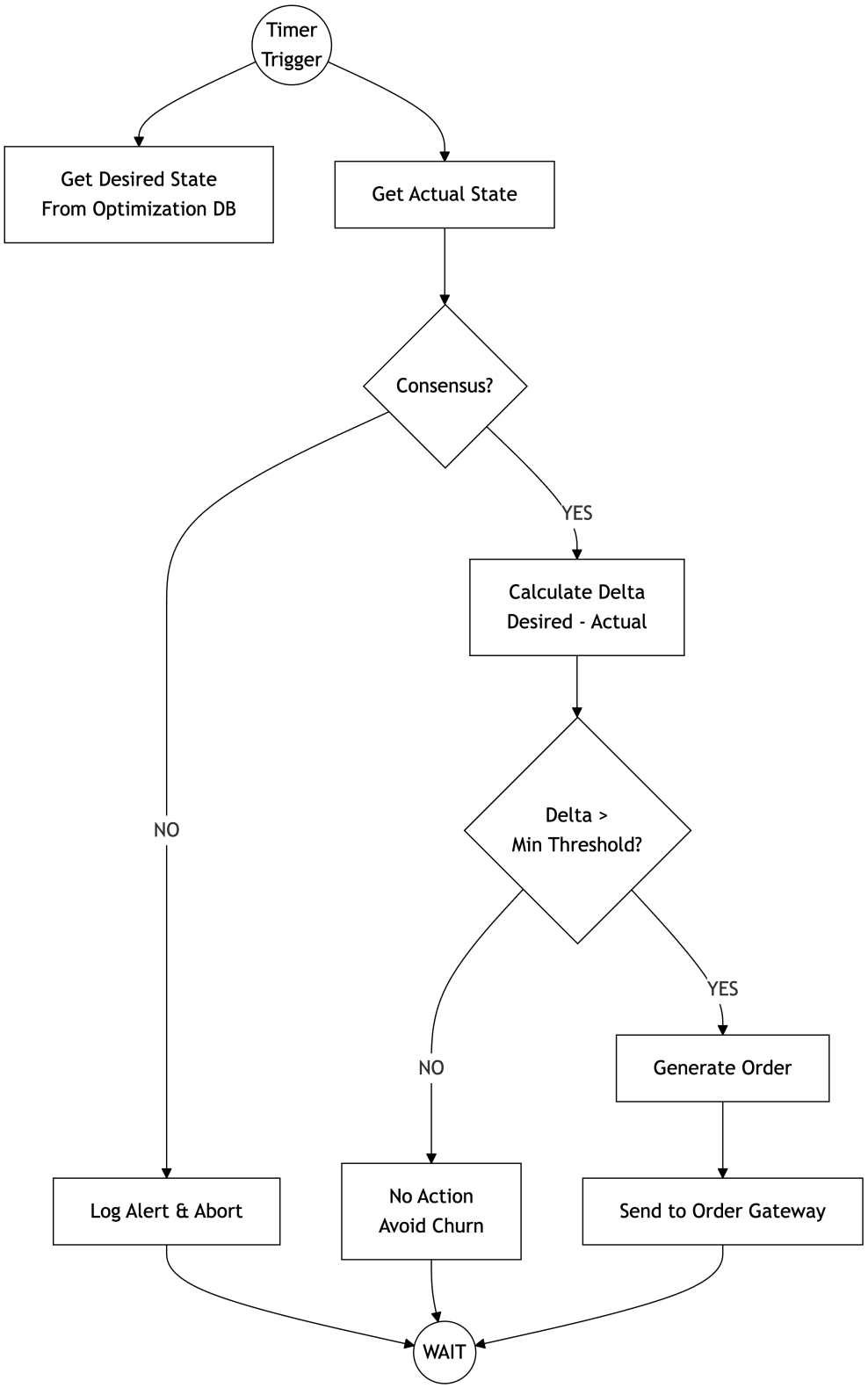

Figure 3: Reconciliation Loop