Strategy Builder & Worker Algorithm Build Process

This document provides a comprehensive flowchart of how the Oculus Strategy system builds trading algorithms, including the role of the LLM (Claude), iteration selection, retraining processes, and how build history influences decisions.

Complete Build Process Flowchart

📊 Strategy Build Process Flowchart

The complete flowchart diagram is available in the following files:

- **PNG Image:** `docs/flowchart.png` - High-resolution flowchart image
- **Mermaid Source:** `docs/flowchart.mmd` - Editable Mermaid diagram source
- **Markdown File:** `docs/STRATEGY_BUILD_FLOWCHART.md` - Interactive version (view in VS Code, GitHub, or any Mermaid-compatible viewer)

💡 **Tip:** Open `flowchart.png` to see the complete color-coded flowchart showing:

- 🔵 **Blue boxes** - LLM operations (Claude Opus 4.6 with Extended Thinking)
- 🟠 **Orange boxes** - ML training and backtesting steps
- 🟣 **Purple boxes** - Scoring and evaluation
- 🟢 **Green boxes** - Final iteration selection

The flowchart visualizes the complete algorithm build process from initial design through iteration loops to final strategy deployment and retraining.

Key Components Explained

1. LLM Role (Claude Opus 4.6 with Extended Thinking)

The LLM plays two critical roles in the build process:

Initial Design Phase (Step 0): - Analyzes build history to learn from past successes/failures - Reviews feature effectiveness data to prioritize indicators - Considers asset class-specific patterns (crypto vs stocks vs forex) - Designs entry/exit rules with specific thresholds - Recommends hyperparameter search space - Suggests model types and neural network architectures - Uses extended thinking mode for deep analysis

Final Review Phase (Optional): - Evaluates all iterations for overfitting risk - Assesses rule consistency across iterations - Recommends the most robust iteration - Identifies potential improvements for future builds

2. Build History Usage

History Scoring System: - **Asset Class Match (+10 points):** Prioritizes builds from same asset class - **Symbol Overlap (+3 points per symbol):** Values builds with similar instruments - **Success Score (+5 points):** Rewards builds with high Sharpe/returns - **Recency Bonus (+0 to +2 points):** Prefers recent builds (within 30 days)

Top 5 builds are selected and their data is fed to the LLM for learning.

3. Iteration Selection Logic

Progressive Complexity: - Each iteration trains with more data (years += 0.25) - Each iteration does more HP tuning (n_iter += 10) - Prevents overfitting by starting simple

Composite Scoring:

```
Score = (Sharpe × 0.30) + (Return × 0.25) + (WinRate × 0.20) +
        (ProfitFactor × 0.15) - (MaxDrawdown × 0.10) + TradeBonus
```

Selection Priority: 1. LLM recommendation (if available) 2. Highest composite score 3. Safety filters applied (min trades, max drawdown, min Sharpe)

4. Retraining Process

When Retraining is Triggered: - Manual user request - Performance degradation detected - Scheduled periodic retrain (e.g., monthly) - Market regime change detected

Retraining Differences: - Loads ALL prior iterations from previous builds - Extracts top-performing features from history - LLM analyzes what worked/didn't work before - Starts with proven configurations - Focuses on incremental improvements

Learning from History: - Feature importance trends across builds - Optimal forward_days and profit_threshold values - Best-performing model types for this strategy - Successful entry/exit rule patterns

5. Worker Job Processing

The worker handles the heavy computational tasks:

1. **Data Fetching:** Multi-source with caching (Polygon → AlphaVantage → Yahoo)
2. **Configuration Search:** Tests multiple label configurations in parallel
3. **Model Training:** Trains 3-5 models simultaneously with HP tuning
4. **Evaluation:** Comprehensive metrics calculation
5. **Rule Extraction:** Converts ML model to interpretable trading rules

6. Decision Points

Critical Decision Points in Flow: - **Retrain vs New Build:** Determines if we use prior iteration history - **Stop Signal Check:** Allows user to abort long-running builds - **Best Iteration Update:** Tracks champion iteration throughout process - **Final Selection:** Chooses deployment-ready iteration with safety checks. LLM reviews the iterations and helps select the best iteration.

Performance Optimization

- **Parallel Processing:** Multiple models train simultaneously
- **Data Caching:** Historical data cached to avoid repeated API calls
- **Progressive Training:** Starts fast with small datasets, scales up
- **Early Stopping:** Can abort if iterations aren't improving

Safety Mechanisms

- **Minimum Trade Count:** Ensures statistical significance (≥10 trades)
- **Maximum Drawdown:** Prevents risky strategies (≤30%)
- **Minimum Sharpe:** Ensures risk-adjusted returns (≥0.5)
- **Overfitting Detection:** LLM reviews for suspicious patterns
- **Rule Consistency:** Validates rules make logical sense