Adaptive Ensemble Strategy for Liquidity Range Management

This research proposes an extension of the tau-reset liquidity strategy for providing liquidity in Uniswap V3. Our approach uses an ensemble of independent predictive models, each forecasting optimal parameters to maximize profitability in the next step. The key innovation lies in a dynamic weighting mechanism that adjusts the importance of individual models within the ensemble.

The strategy works as follows:

1. Collection of predictions from multiple models regarding position adjustments
2. Calculation of a weighted average decision based on each model's assigned importance
3. Periodic update of model weights using gradient descent based on observed performance

This methodology aims to answer two key research questions:

- Can such an ensemble reach a stable state with fixed weight coefficients?
- Will this approach reveal the relative effectiveness of different predictive models under various market conditions?

Additionally, we investigate the minimum future price window required for the reference strategy to provide optimal results for weight adjustment.

1 Operational Framework

Below are the main stages of the strategy's operation

Stage 1: For step n, we obtain actions from all models a_1, a_2, a_3 and aggregate them into a single prediction d_n.

Stage 2: At step d_a, we know the prices from that step until the present moment. We calculate the optimal strategy for the step a using the reference strategy ($\text{Oracle}(s_a, \ldots, s_n) = d_a^o$), obtain the result d_a^o, and adjust the model weights. w is the delay between the correction step and the present moment.

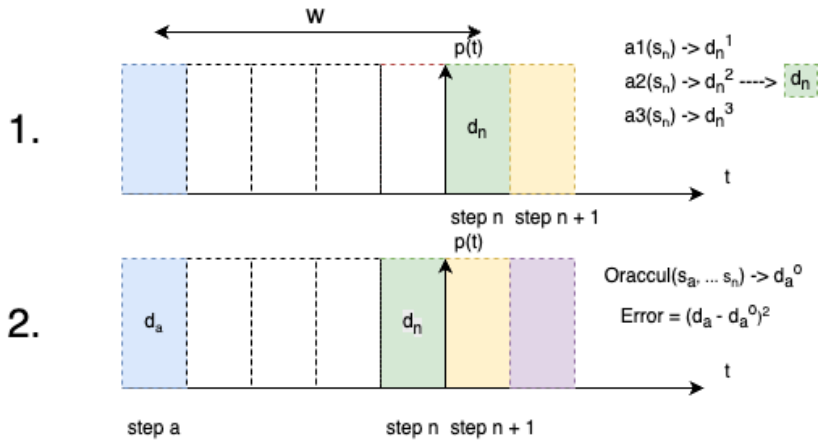

Figure 1: Strategy operational stages

2 Core System Components

2.1 Decision Vector

Each model i at time t outputs a prediction vector:

$$\mathbf{s}_i^{(t)} = [r_i^{(t)}, c_i^{(t)}, w_i^{(t)}]^T$$

where:

- $r_i^{(t)} \in [0, 1]$ - probability of rebalancing necessity
- $c_i^{(t)} \in \mathbb{R}$ - predicted range center
- $w_i^{(t)} \in \mathbb{R}^+$ - predicted range width

Thus, a value of $r = 0.0$ means that the position should be removed if it exists, and nothing should be done if it doesn't. A value of $r \neq 0$ and $c = c(n-1)$ means that the current position should not be changed. A value of $r \neq 0$ and $c \neq c(n-1)$ means that a rebalancing should be performed if a position exists, or a position should be added if it doesn't.

2.2 Weighted Voting

The final ensemble decision:

$$\mathbf{S}^{(t)} = \frac{\sum_{i=1}^{N} \alpha_i^{(t)} \cdot \mathbf{s}_i^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

where $\alpha_i^{(t)}$ is the weight of model i at time t.
Component-wise:

$$R^{(t)} = \frac{\sum_{i=1}^{N} \alpha_i^{(t)} \cdot r_i^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}; \quad C^{(t)} = \frac{\sum_{i=1}^{N} \alpha_i^{(t)} \cdot c_i^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}; \quad W^{(t)} = \frac{\sum_{i=1}^{N} \alpha_i^{(t)} \cdot w_i^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

3 Delayed Learning

3.1 Optimal Solution

After a delay time τ, we calculate the optimal solution:

$$\mathbf{S}_{opt}^{(t)} = [R_{opt}^{(t)}, C_{opt}^{(t)}, W_{opt}^{(t)}]^T$$

where the optimal values are determined through the reference strategy.

3.2 Loss Function

For each component of the decision vector:

$$L_r = (R^{(t)} - R_{opt}^{(t)})^2$$

$$L_c = (C^{(t)} - C_{opt}^{(t)})^2$$

$$L_w = (W^{(t)} - W_{opt}^{(t)})^2$$

4 Gradient Descent for Weight Updates

4.1 Weight Gradients

For component r (similarly for c and w):

$$\frac{\partial L_r}{\partial \alpha_j} = 2(R^{(t)} - R_{opt}^{(t)}) \cdot \frac{\partial R^{(t)}}{\partial \alpha_j}$$

where:

$$\frac{\partial R^{(t)}}{\partial \alpha_j} = \frac{r_j^{(t)} \sum_{i=1}^{N} \alpha_i^{(t)} - \sum_{i=1}^{N} \alpha_i^{(t)} r_i^{(t)}}{(\sum_{i=1}^{N} \alpha_i^{(t)})^2} = \frac{r_j^{(t)} - R^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

Therefore:

$$\frac{\partial L_r}{\partial \alpha_j} = 2(R^{(t)} - R_{opt}^{(t)}) \cdot \frac{r_j^{(t)} - R^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

5 Relative Weight Updates

5.1 Delta Calculation

Weight change for model j:

$$\Delta \alpha_j = -\eta \frac{\partial L}{\partial \alpha_j}$$

where η is the learning rate.

5.2 Mean Normalization

We calculate the average change:

$$\overline{\Delta \alpha} = \frac{1}{N} \sum_{i=1}^{N} \Delta \alpha_i$$

We adjust the deltas to create competition:

$$\Delta \alpha_j^{adj} = \Delta \alpha_j - \overline{\Delta \alpha}$$

5.3 Weight Updates

Updated weights:

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} + \Delta \alpha_j^{adj}$$

6 Complete Learning Algorithm

Initialization:

- Set initial weights: $\alpha_i^{(0)} = 1/N$ for all i
- Set parameters: $\eta, \tau, \alpha_{min}, \alpha_{max}$

Main Loop:
For each time t:

1. **Prediction Collection:**

$$\mathbf{s}_i^{(t)} = \text{Model}_i(\text{MarketData}^{(t)})$$

2. **Decision Aggregation:**

$$\mathbf{S}^{(t)} = \frac{\sum_{i=1}^{N} \alpha_i^{(t)} \cdot \mathbf{s}_i^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

3. **Decision Making:**

   - If $R^{(t)} > \theta$: perform rebalancing with parameters $(C^{(t)}, W^{(t)})$
   - Otherwise: maintain the current position

4. **Learning (at time $t + \tau$):**

   a. Calculate the optimal solution using the reference strategy

   b. Calculate gradients:

$$\frac{\partial L}{\partial \alpha_j} = \sum_{k \in \{r,c,w\}} \lambda_k \cdot 2(K^{(t)} - K_{opt}^{(t)}) \cdot \frac{k_j^{(t)} - K^{(t)}}{\sum_{i=1}^{N} \alpha_i^{(t)}}$$

   c. Calculate weight changes:

$$\Delta\alpha_j = -\eta \frac{\partial L}{\partial \alpha_j}$$

   d. Normalize relative to the mean:

$$\Delta\alpha_j^{adj} = \Delta\alpha_j - \frac{1}{N} \sum_{i=1}^{N} \Delta\alpha_i$$

   e. Update weights:

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} + \Delta\alpha_j^{adj}$$

7 Implementation Process

The work was based on the original tau-reset strategy code in the fractal-defi repository.

7.1 Adding the Ability to Not Hold a Position

This functionality was added to the original strategy code based on the aforementioned rules about cases of maintaining a position in the same place

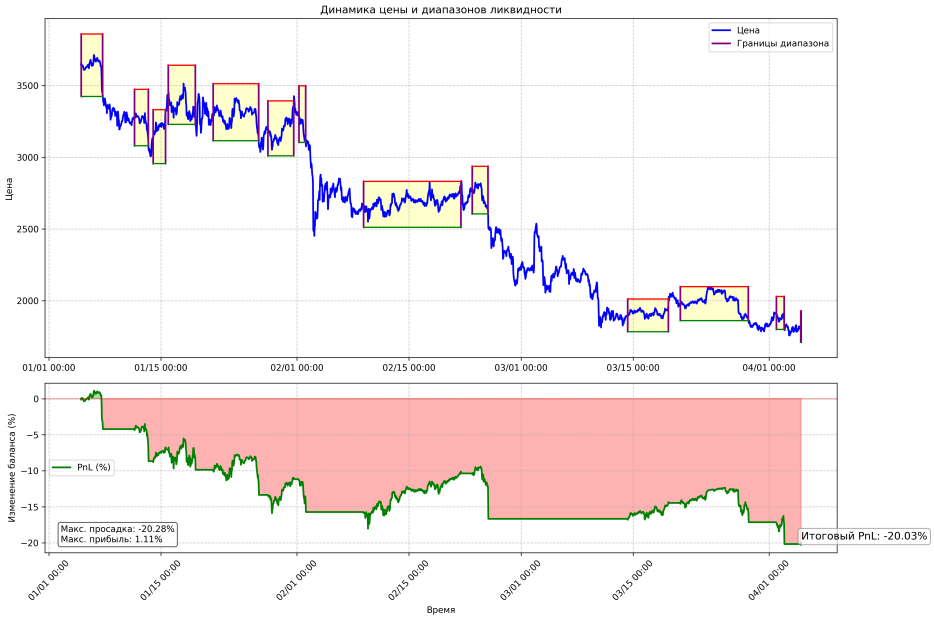

Figure 2: Strategy with intervals between positions

7.2 Creating the Reference Algorithm

Since the algorithm will process past history and rely on an already known price distribution, we can say that this algorithm "knows the near future" and uses it for decision-making. It is important to mention that the less this algorithm needs to know about the future to implement a good strategy, the better, as the adjustment at step n will be based on step $(n - \text{window\_size})$. Considering all factors for creating such an algorithm is very complex, so the main objective of the algorithm was chosen to be the minimization of IL (Impermanent Loss).

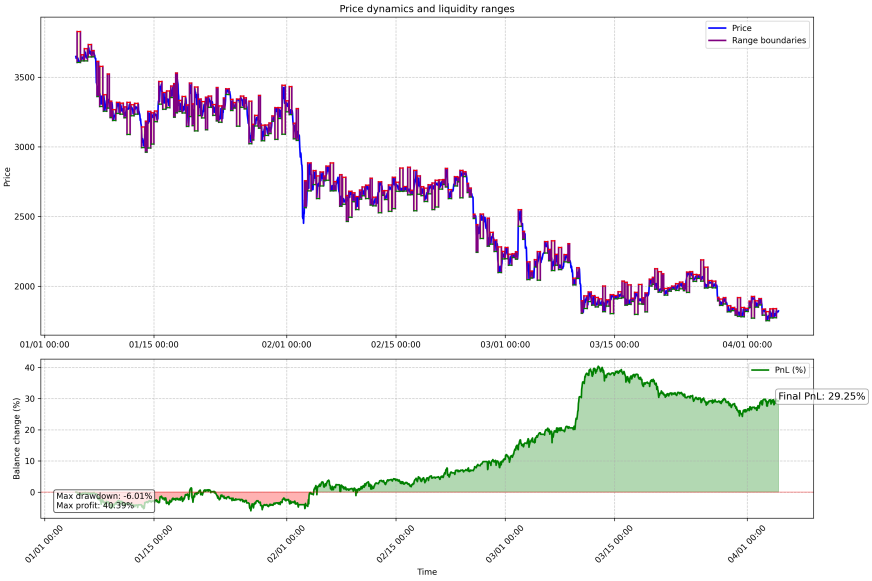

Figure 3: One of the results of the reference strategy operation

Below is a table with various future price window sizes and the total income after strategy implementation as a percentage of the initial balance. The minimum position length was 5 ticks.

Future Price Window Size (ticks)	Percentage of Initial Income (%)
30	55.00
25	45.54
20	50.18
15	32.21
13	31.95
12	29.25
11	18.02
10	3.95
9	0.74
8	-8.73
7	-13.10
6	-21.84

Table 1: Dependency of strategy efficiency on future price window size

8 Ensemble Algorithm for Strategy Combination

To demonstrate the algorithm's operation and test weight adjustments, we selected an ensemble of classical tau-reset strategies. This strategy maintains a constant position and performs rebalancing when the price exceeds the range boundaries. The range size is fixed, with algorithms differing only in their range sizes. This approach effectively creates an algorithm with dynamic tau parameter adjustment based on historical values.

8.1 Experimental Results

Below, we describe experiments combining two algorithms with $\tau = 5$ and $\tau = 30$. For this combination, the only parameter receiving feedback and adjustment is the range size (W).

Figure 4 shows how the coefficients converge to a specific value with minor fluctuations around it.

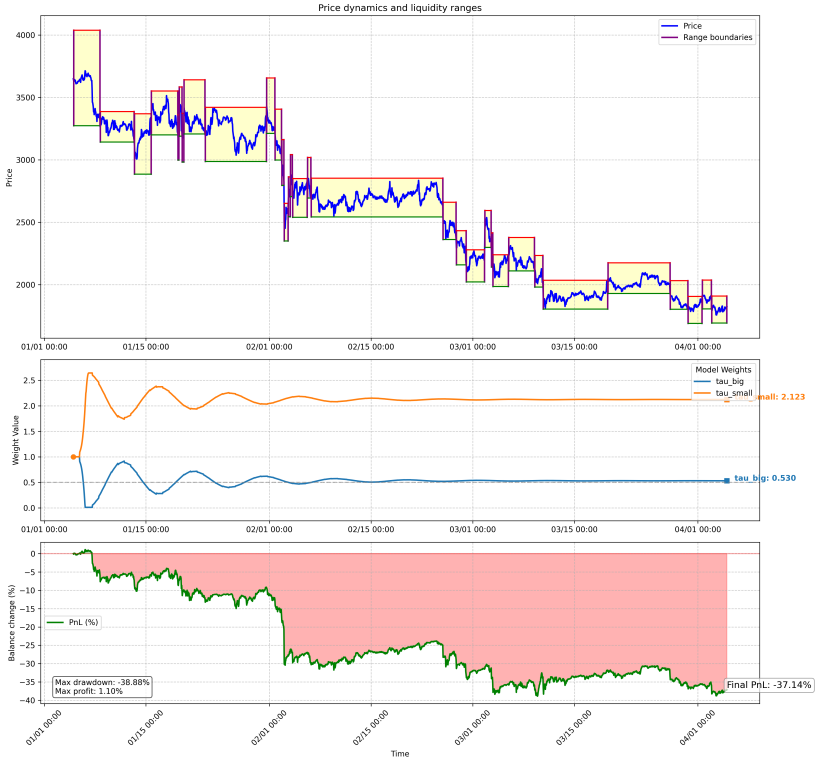

Figure 4: Convergence of model weights over time

To verify that the coefficients not only converge but actually adjust to increase profitability, we compared this ensemble with a single classical tau strategy where τ equals the mean value that our ensemble appears to converge to. The simulation was performed using data from the ETHUSDT pool on Ethereum Mainnet for the period from January 5, 2024, to April 5, 2024.

Figure 5 compares these two strategies.

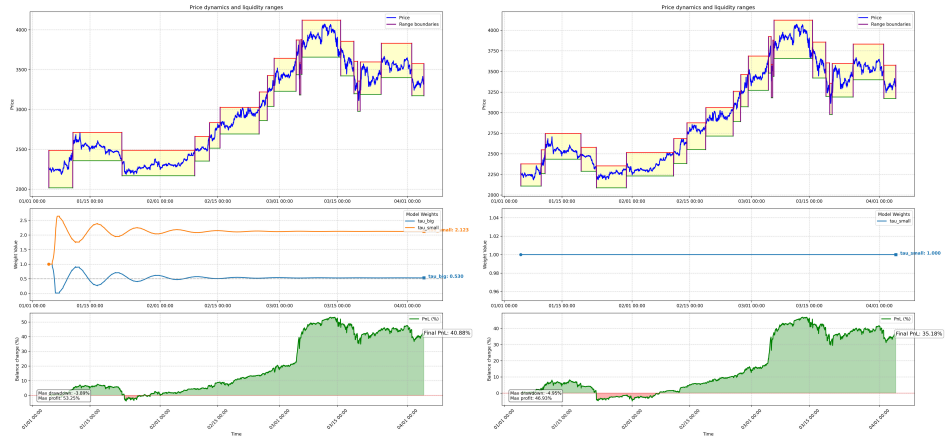

Figure 5: Performance comparison between the ensemble (left) and fixed-τ strategy (right)

The results clearly demonstrate superior profitability compared to the fixed-τ strategy, with a 5.7% increase in profit during the test period. Thus, we have created an adaptive τ strategy without explicitly defining how it should change, relying solely on the reference algorithm for guidance.

9 Conclusions

We have successfully implemented an adaptive ensemble system for optimal liquidity provision on Uniswap V3. The model can be easily narrowed down for smaller experiments, such as the adaptive tau task demonstrated earlier. The performance quality depends on the algorithms used and primarily on the reference algorithm: how effectively it maximizes profit and how many future steps it requires to do so.

The current reference algorithm does not account for numerous factors and requires further refinement. Future work could focus on incorporating more sophisticated base models and developing reference algorithms that consider additional factors such as gas costs, market impact, and counterparty behavior.