Scaled Beta Policies for Uniswap V3 Liquidity Pools

Prior Work (Focused on Traditional Market Making with Central Limit Order Books)

Yongzhao Wang, Rahul Savani, Anri Gu, Chris Mascioli, Theodore Turocy, and Michael Wellman. 2024. ***Market Making with Learned Beta Policies***. In *Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF '24),* November 14–17, 2024. https://doi.org/10.1145/3677052.3698623

Current Work:

Can Scaled Beta Policies Efficiently Allocate Capital in Uniswap V3 Liquidity Pools?

Sam Gabor, sg662, IEORE8100

Overview

BACKGROUND

METHODOLOGY

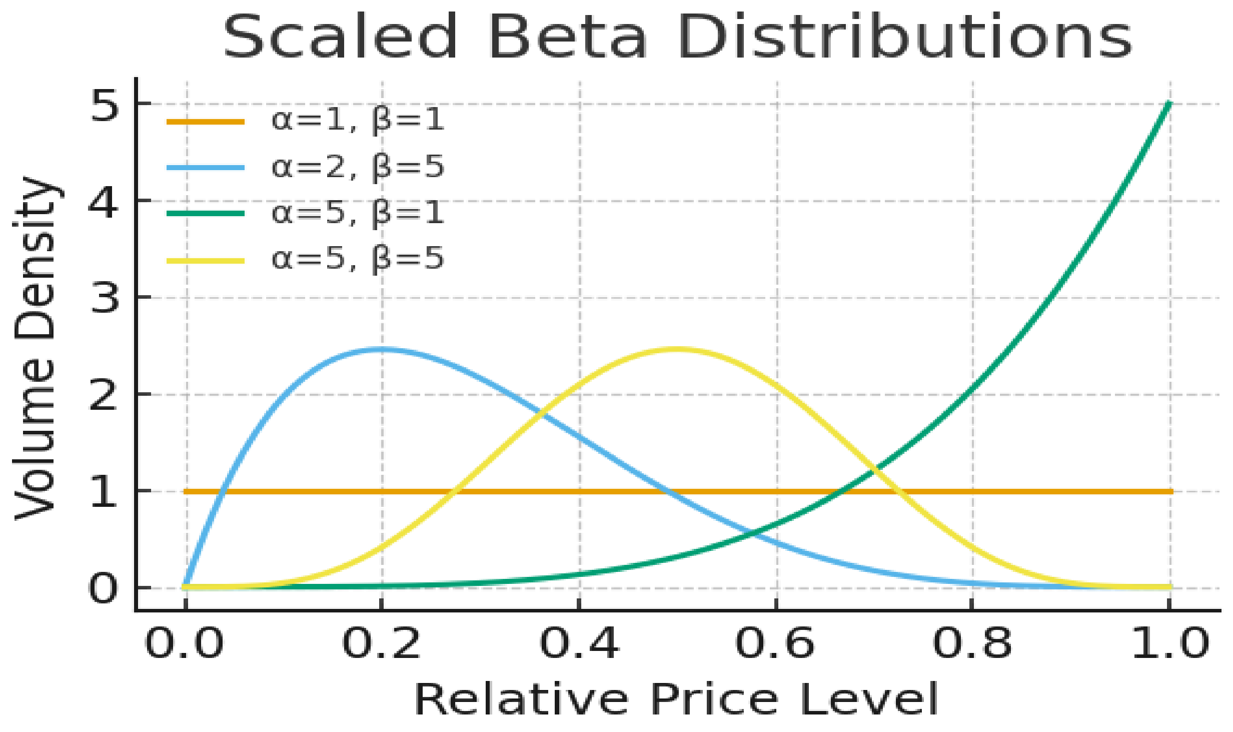
RESULTS

CONCLUSIONS

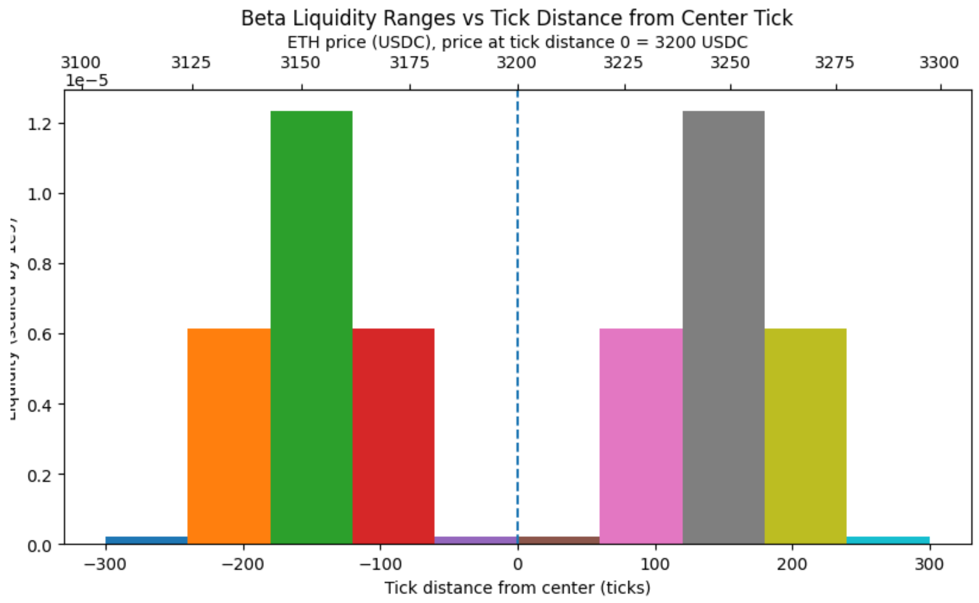
Scaled Beta Distribution

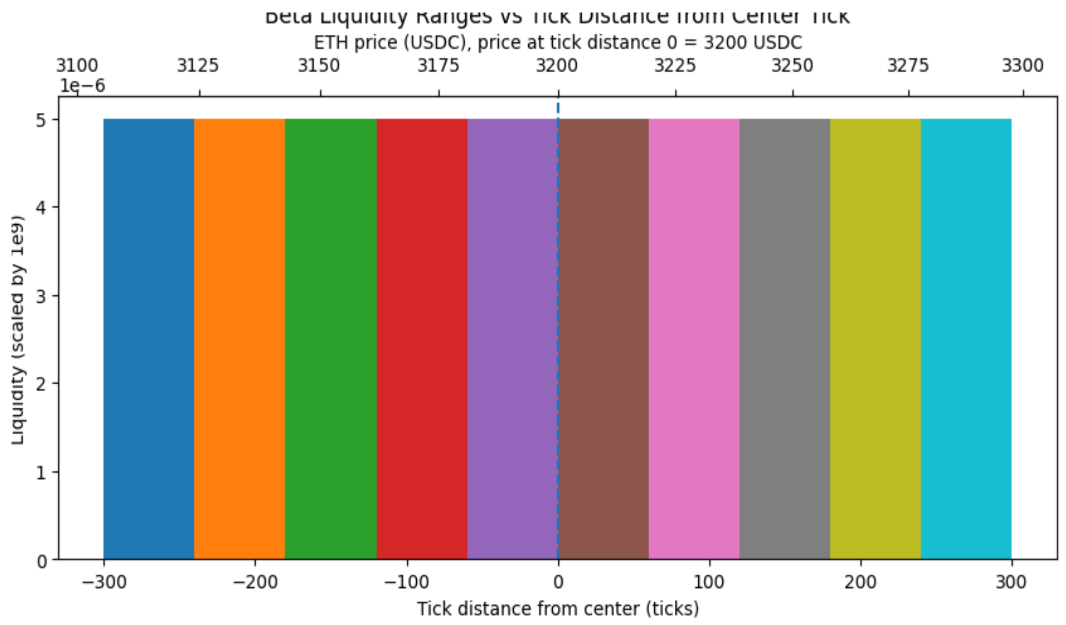
- Scaled Beta Distribution can be shaped by setting different values for **α** and **β**
- Shape can be used to distribute order volumes in CLOBs or AMM liquidity depths in price ranges

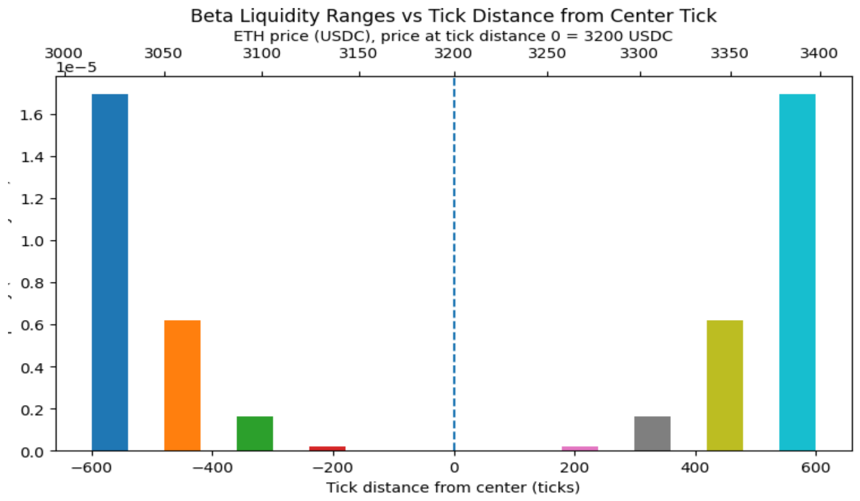
$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt}$$



Scaled Beta Distribution – Simulation Examples







Methodology – Overall Approach

Run a simulation with order flow, stochastic ETH pricing and arbitrage to compare the effects of different Beta Distribution parameters applied to LPs on the profit and loss of a portfolio of 20 ETH/50K USDC.

Methodology – Tools

Simulation uses:

- Anvil, a local in-memory Ethereum blockchain node
- Forge, a smart-contract development toolchain for Ethereum
- Deployed on-chain contracts (Forked Uniswap V3 artifacts)
- Off-chain python simulation driver (interacts with Anvil using web3)

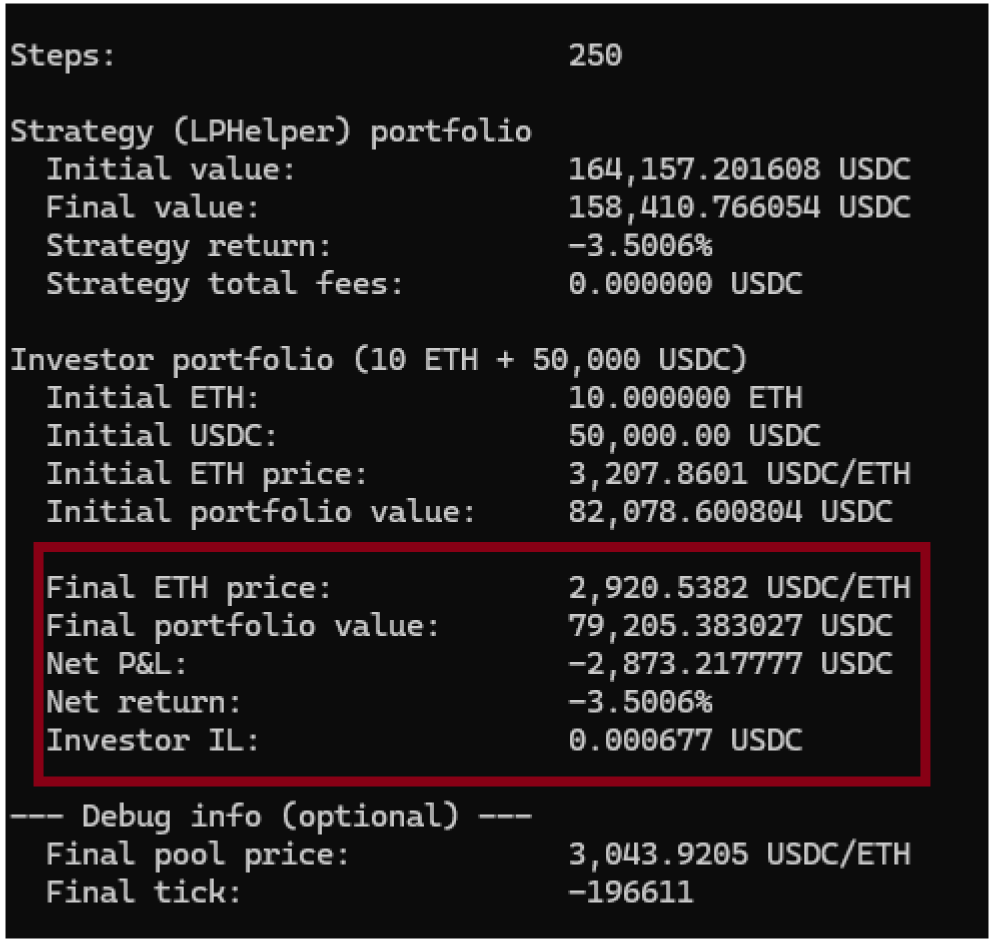
Methodology – Custom Solidity Components

- **MockERC20 (WETH + USDC)**
  - Deterministic unlimited minting of WETH and USDC
  - Seeds tokens for simulation
- **LPHelper.sol**
  - Mints beta-ranges
  - Pokes positions to realize fees
  - Computes Uniswap V3 position principal + fees
  - Aggregates liquidity across all configured ranges
  - Wraps all Uniswap V3 math (TickMath, LiquidityAmounts, slots, sqrtPrice, etc.)
- **SwapHelper.sol**
  - Executes simulated swaps
  - Provides entry point for off-chain engine
  - Supports buy/sell WETH order flow
- **Bootstrap Scripts (.s.sol)**
  - Deploys tokens
  - Deploys pool (100% Uniswap V3 compliant factory, positions, pool)
  - Seeds balances in LPHelper + SwapHelper

Methodology – Custom Python Components

- **run_orderflow.py**
  - Drives the entire simulation
  - Sends swaps (trades) to SwapHelper
  - Performs arbitrage (optional)
  - Records state after each step
  - Computes fee growth, principal, IL
  - Records portfolio into episode.csv
- **sim_step.py**
  - Executes a single simulation step
  - Applies Ornstein–Uhlenbeck(OU) [mean reverting stochastic] process for ETH price changes
  - Sends trade to on-chain SwapHelper
  - Loops arbitrage until pool price ≈ market price
- **state_snapshot.py**
  - Reads on-chain Uniswap V3 storage
  - Computes position principal using Uniswap math
  - Computes uncollected fees (via pokeRanges)
  - Returns portfolio composition in token units
- **summarize_episode.py**
  - Computes investor P&L
  - Computes investor IL
  - Computes investor fee income
  - Summarizes strategy performance

Results: Sample Episode Summary



```
Steps:                             250

Strategy (LPHelper) portfolio
  Initial value:                   164,157.201608 USDC
  Final value:                     158,410.766054 USDC
  Strategy return:                 -3.5006%
  Strategy total fees:             0.000000 USDC

Investor portfolio (10 ETH + 50,000 USDC)
  Initial ETH:                     10.000000 ETH
  Initial USDC:                    50,000.00 USDC
  Initial ETH price:               3,207.8601 USDC/ETH
  Initial portfolio value:         82,078.600804 USDC

  Final ETH price:                 2,920.5382 USDC/ETH
  Final portfolio value:           79,205.383027 USDC
  Net P&L:                         -2,873.217777 USDC
  Net return:                      -3.5006%
  Investor IL:                     0.000677 USDC

--- Debug info (optional) ---
  Final pool price:                3,043.9205 USDC/ETH
  Final tick:                      -196611
```

Results: What the Simulation Revealed

Simulating Uniswap V3 with high fidelity is *hard!*

- Beta ranges behaved consistently from a P&L perspective across α and β parameters. [No standout parameter values]
- Fee income was very small
- IL can be better managed with symmetrical ranges but not eliminated.
- Arbitrage was essential to realize correct tick balances.

Results: What Requires More Research

- Fees were almost zero in many runs because:
  - price didn’t cross into many beta ranges, or
  - we lacked a persistent arbitrageur touching all ranges.
- Impermanent loss often appeared tiny due to:
  - symmetric ranges
  - price finishing near the start
  - low realized volatility

Conclusions and Demo

Can scaled beta policies help manage P&L in Uniswap V3 LPs? Yes but only modestly:

- Beta ranges do smooth IL over price movement
- Diversifying ranges reduces extreme outcomes
- Fee income appears tiny (needs further research to determine cause)

In realistic settings, beta policies help but do not solve P&L instability, and their performance may entirely be dictated by other market factors (MEV, etc.) rather than the range scheme explored.

Questions?