Bandit Funding Rates

team@vest.xyz

December 2022

Introductory Notes

What follows below is research carried out by Vest Labs with the primary intent of it being discussed on the labs' forum. This particular session can be found here: https://research.vest.xyz/t/bandit-funding-rates/39

Motivation

In our previous post *New Framework for Funding Rates,* we investigated the current landscape of funding rates for perpetual futures and explored two new approaches: risk-based and adaptive. For this write-up, we look to unify both approaches by inheriting a profit maximization objective and formalizing setting funding rates as an adversarial multi-armed bandit problem.

Problem Formalization

Since AMM-based exchanges receive funding payments due to a long-short imbalance, we frame the problem of setting variable funding rates similarly to that of a market maker setting bid-ask spreads to maximize profit.

Let L_t, $S_t \in \mathbb{R}^+$ be long, short open interest (in number of contracts) at time t respectively. For terminal time T, define cumulative realized PnL as $RPnL_T := F_T^t + F_T^f + X_T$ where F_T^t is cumulative trading fees received, F_T^f is cumulative excess funding received (funding paid from the overweight side to the exchange), and X_T is realized PnL against traders due to the AMM acting as the counterparty against traders. Define total unrealized PnL as $UPnL_T := -P_T^{liq}(L_T - S_T)$ where P_T^{liq} is the liquidation price of the imbalance liability from the long-short imbalance. Hence, our objective function is

$$\text{maximize} \quad RPnL_T + UPnL_T$$

We choose a reinforcement learning (RL) approach and apply a multi-armed bandit (MAB) framework due to the computational constraints of blockchain. In particular, we use adversarial bandits for our highly dynamic context.

Arms

Consider each round as a time t. Define realized PnL per round as

$$\Delta RPnL_{t+1} := P_t f_t^t V_t + P_t f_t^f (L_t - S_t)$$

where f_t^t is trading fee (%), V_t is trading volume, f_t^f is the funding rate (%), and P_t is the index price. Define unrealized PnL per round as

$$\Delta UPnL_{t+1} := -(\Delta P_{t+1})(L_t - S_t)$$

where $\Delta P_{t+1} := P_{t+1} - P_t$. Note that $\sum_{t=1}^{T}(\Delta RPnL_t + \Delta UPnL_t) = RPnL_T + UPnL_T$.

To reduce model risk, we utilize observable metrics and choose the parametrization for desired funding received by the exchange per round to be

$$F_t^f = -\frac{1}{\alpha\lambda_t} \cdot \min(0, \sum_{s=1}^{t}(\alpha\Delta RPnL_s + \Delta UPnL_s))$$

where $\alpha \in (0, 1]$. Intuitively, we assume that $(1 - \alpha)$ of the realized PnL is set aside for external LPs and set λ_t as our arm for bandit algorithms, essentially choosing the rate at which any potential loss is covered with α of funding payments (see Appendix A.1 for the derivation). During each round, the choice of an arm is constrained to the subset of arms that do not result in a funding rate $f_t^f \geq 1$, which would charge traders more funding than their total collateral. Theoretical analysis of regret bound under such constrained algorithm is left for future work. Note that we refer to this arm parametrization as Risk Decay(α) in our simulation code.

Reward Function

We consider asymmetrically dampened PnL as our reward function, which is used in a number of previous works on market making with reinforcement learning [6]. More precisely, we define our reward function as the sum of realized PnL and unrealized PnL with an additional penalty term on positive unrealized PnL:

$$r_t = \Delta RPnL_t + \Delta UPnL_t - \eta \cdot \max(0, \Delta UPnL_t)$$

where $\eta \in [0, 1]$. As η increases, we add more penalty for positive unrealized PnL while keeping negative unrealized PnL intact, thereby discouraging the AMM from gaining profits through speculation by holding a long-short imbalance. In practice, the reward must be normalized to $\tilde{r}_t$ such that $\tilde{r}_t \in [0, 1]$ (see Appendix A.2 for more details). Note that we refer to this reward function as RwDU(η), Realized PnL with Dampened Unrealized PnL, in our simulation code.

Policy

We employ multiple adversarial bandit algorithms used in the literature: ε-greedy [9], Follow-the-Perturbed-Leader (FPL) [2], Exp3 [3], and Boltzmann-Gumbel [5] (a variant of exponential weight algorithm). Given normalized reward $\tilde{r}_t$, each algorithm updates weights assigned to each arm i from p_t^i to p_{t+1}^i and select the next arm using these weights. We also introduce Static strategy, where the algorithm chooses the fixed arm regardless of the reward.

While Exp3 is commonly used for adversarial bandit problems due to its sublinear regret bound (performance loss against the best arm in hindsight), we explore other algorithms as well since practical performance tends to depend on the context. In particular, multiple studies have shown that ε-greedy algorithm, despite having a linear regret bound with a constant ε, produces comparable empirical results and is worth simulating [9].

Simulation

We construct an agent-based model (ABM) and perform Monte Carlo simulations to examine the PnL profiles of different bandit parametrizations. Inspired by the Lux-Marchesi multi-agent model [7], we include 3 agents in our ABM: chartists (c), fundamentalists (f), and funding rate arbitrageurs (a). Each agent i with $type \in \{c, f, a\}$ is endowed with wealth $w_i^{type} \sim \text{Pareto}(1, \alpha)$ and assigned a time horizon $h_i^{type} \sim U(1, h_{max}^{type})$ with granularity g_i^{type}, which they use as the lookback window length to calculate moving averages.

Agents

1. **Chartists** are speculative agents who trade on price trend. Chartists will buy if $P_t > P_t^{ma}$ and sell otherwise, where $P_t^{ma} = \frac{1}{h^c}\sum_{s=t-h^c+1}^{t} P_s$ is the moving average price at time t. Chartists size their orders $\propto \frac{w^c|P_t - P_t^{ma}|}{P_t}$ in number of contracts.

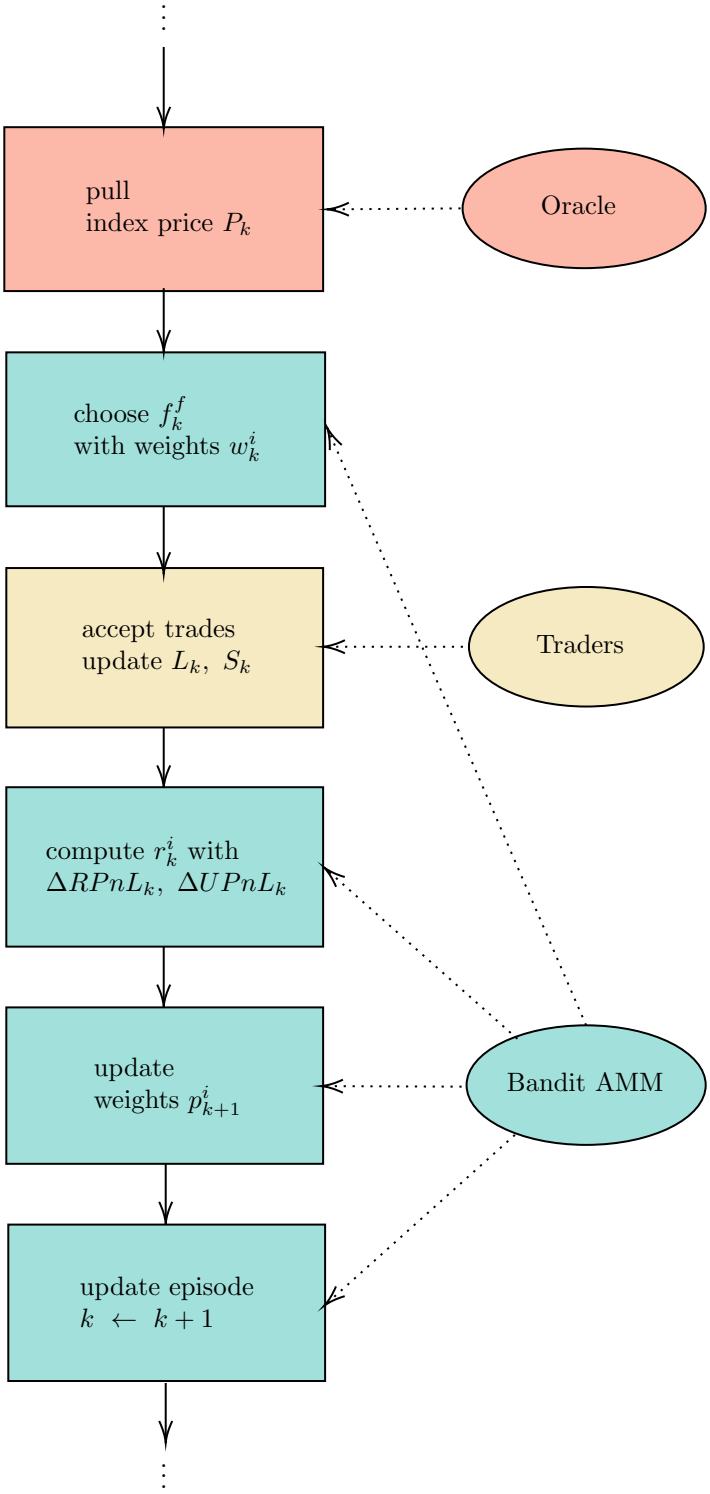

Figure 1: Flowchart of Bandit AMM

2. **Fundamentalists** are stabilizing agents who expect the price to revert to its fundamental value. We capture the fundamental value by a moving average price with larger granularity $g^f > g^c$ (hence longer time horizons) than chartists. Fundamentalists will buy if $P_t < P_t^{ma}$ and sell otherwise, sizing their orders similarly to chartists.

3. Funding rate arbitrageurs (henceforth **arbitrageurs**) receive funding by helping to correct the long-short imbalance while remaining delta-neutral. Arbitrageurs observe a moving average of the funding rate f_t^{ma} and are assigned a market friction $fr \sim U(0, fr_{max})$. If the expected funding received is greater than market friction, i.e. $f_t^{ma} \cdot h^f > fr$, then they will open a position in the direction of receiving funding (buy if $f_t^{ma} < 0$, sell otherwise). They will close their position if $f_t^{ma} \cdot h^f \leq fr$. Arbitrageurs size their orders $\propto f_t^{ma} \cdot h^f - fr$ in number of contracts.

During each round of the simulation, the AMM selects a funding rate via the bandit algorithm after observing a reward from the previous round. The AMM charges funding from traders who have positions open and then proceeds to accept new orders. Agents with negative or zero wealth are removed from the simulation. See Figure 1 for a flowchart of the AMM.

Importantly, both chartists and fundamentalists do not make trading decisions based on the funding rate, but are nonetheless charged funding. If funding rate remains overly high for a long period of time, these agents will have less wealth, thereby reducing trading volume and hence the exchange's fee revenue.

Results

To keep our calibration procedure simple, we assume an exogenous price time series and look to replicate trading activity (see details in Appendix B.1). We used grid search and conducted Monte Carlo simulations to obtain the following optimal parametrization:

α	η	Policy
0.1	0.5	Exp3

See Figure 2 for results of a single run with the above parametrization and Figure 4 for a comparison of the different parametrizations.

(See our simulation code here)

Discussion

For the future, there are several limitations that can be addressed. First, a profit maximization objective seems to be more fitted to a market maker, rather than an exchange. This is clear from the simulated high trading volume despite high funding rates due to funding rate arbitrageurs, reminiscent of The Perpetual PvP Ponzi. Instead, a welfare maximization perspective with an objective of minimizing both cost of funding and insolvency risk can be the next framework to explore.

High funding rates also motivated our approach in constraining the choice of arms, which should also be investigated further in terms of theoretical regret bounds and empirical performance. Importantly, there remains a problematic edge case when funding charged exceeds the overweight sides' total collateral and unrealized PnL, resulting in insolvency. It follows that additional guard rails, such as open interest caps, may have to be implemented to complement a MAB framework.

The calibration and validation of our ABM can also be improved in various ways. We observed that if the exchange happened to hold a favorable imbalance liability, the resulting PnL would be tremendously positive. To combat this, chartists and fundamentalists could be more intelligent by refusing to trade when the funding rate is unreasonably high. Alternatively, we can also use price paths on other dates for simulations.

In terms of choosing the best bandit algorithm, it seems that the optimal algorithm is rather context dependent. While our current approach is to have a single optimization algorithm choosing an appropriate

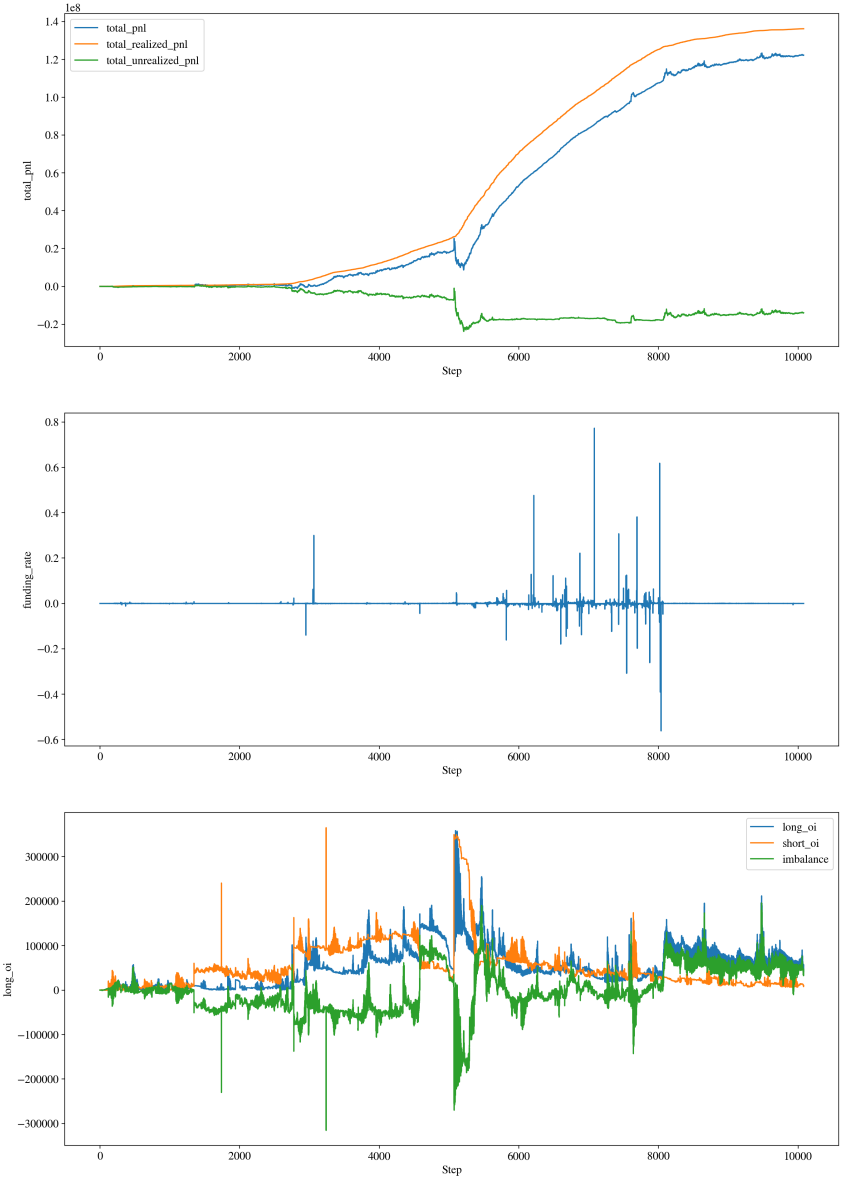

Figure 2: Single run results ($\alpha = 0.1, \eta = 0.5$, Policy = Exp3)

arm (funding rate) to employ, it is of interest to explore how this can be extended to more decentralized setting (in a similar way as how Aera [1] aggregates portfolio submissions).

References

[1] Aera Protocol. Risk-Aware DAO Treasury Management. 2022.

[2] Jacob Abernethy and Satyen Kale. Adaptive Market Making via Online Learning. 2015.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freunds, and Robert E. Schapire. The Nonstochastic Multi-Armed Bandit Problem. 2002.

[4] Guus ten Broeke, George van Voorn and Arend Ligtenberg. Which Sensitivity Analysis Method Should I Use for My Agent-Based Model? 2016.

[5] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugos and Gergely Neu. Boltzmann Exploration Done Right. 2017.

[6] Bruno Gašperov, Stjepan Begušić, Petra Posedel Šimović and Zvonko Kostanjčar. Reinforcement Learning Approaches to Optimal Market Making. 2021.

[7] Thomas Lux and Michele Marchesi. Scaling and criticality in a stochastic multi-agent model of a financial market. 1999.

[8] Magnus Erik Hvass Pedersen. Good Parameters for Particle Swarm Optimization. 2010.

[9] Joannes Vermorel and Mehryar Mohri. Multi-armed Bandit Algorithms and Empirical Evaluation. 2005.

Appendix A Bandit Framework Specifics

A.1 Arms

At a high level, we want the imbalance risk at time t to decay to zero over time by incentivizing funding rate arbitrageurs to correct the long-short imbalance or otherwise by receiving excess funding. Using unrealized loss as a proxy for imbalance risk, at some future time $t + \lambda_t$ and assuming we set aside $(1 - \alpha)$ of any realized PnL perhaps as yield for external LPs, we want to find the smallest F_t^f such that $\sum_{s=1}^{t+\lambda_t} \alpha \Delta RPnL_s + \Delta UPnL_s \geq 0$. Then, we have

$$\sum_{s=1}^{t+\lambda_t} \alpha \Delta RPnL_s + \Delta UPnL_s = \sum_{s=t+1}^{t+\lambda_t} \alpha \Delta RPnL_s + (\sum_{s=1}^{t} \alpha \Delta RPnL_s + \Delta UPnL_s) \geq 0$$

$$\iff \sum_{s=t+1}^{t+\lambda_t} \alpha \Delta RPnL_s \geq \sum_{s=1}^{t} (\alpha \Delta RPnL_t + \Delta UPnL_t)$$

$$\iff \lambda_t \cdot \alpha \cdot F_t^f = -\min(0, \sum_{s=1}^{t} (\alpha \Delta RPnL_t + \Delta UPnL_t))$$

$$\iff F_t^f = -\frac{1}{\alpha \lambda_t} \cdot \min(0, \sum_{s=1}^{t} (\alpha \Delta RPnL_s + \Delta UPnL_s))$$

where F_t^f is desired funding payment received per round. Note that we have assumed there to be no extra trades between $t + 1$ and $t + \lambda_t$ (realized PnL does not include any trading fees) and that unrealized PnL remains the same, i.e. $\sum_{s=t+1}^{t+\lambda_t} \Delta UPnL_s = 0$.

A.2 Reward Function

In order to normalize the reward $r_t = \Delta RPnL_t + \Delta UPnL_t - \eta \cdot \max(0, \Delta UPnL_t)$, we must approximate the bounds for $\Delta RPnL_t$ and $\Delta UPnL_t$. Here,

$$\begin{aligned} \Delta RPnL_t &= P_t f_t^t V_t + P_t f_t^f (L_t - S_t) \\ &< P_t (f_t^t + f_t^f)(L_t + S_t) \quad \text{(assuming } V_t < L_t + S_t \text{ with high probability)} \\ &< P_t (L_t + S_t) \quad \text{(assuming } f_t^t + f_t^f < 1 \text{ with high probability)} \end{aligned}$$

and

$$\begin{aligned} |\Delta UPnL_t| &= |\Delta P_{t+1} (L_t - S_t)| \\ &\leq |\Delta P_{t+1} (L_t + S_t)| \\ &< |P_t (L_t + S_t)| \quad (P_{t+1} \in (0, 2P_t] \text{ with high probability}) \end{aligned}$$

Then, $\Delta UPnL_t - \eta \cdot \max(0, \Delta UPnL_t) \in [-P_t(L_t+S_t), (1-\eta)P_t(L_t+S_t)]$ with high probability. Putting these all together and letting $TN_t := P_t(L_t + S_t)$, defining normalized reward $\tilde{r}_t := \frac{r_t + TN_t}{(3-\eta)TN_t}$ leads to $\tilde{r}_t \in [0, 1]$ with high probability. Given these are somewhat loose bounds for $\Delta RPnL_t$ and $\Delta UPnL_t$, it is possible that the rewards are normalized down to $[0, \alpha]$ with $\alpha \ll 1$ and that the algorithms are less sensitive to the difference in performance among the arms. It is of further interest to exploring other reward functions with more accurate bounds.

A.3 Policy

Below are the parameters used for each of the algorithms. Note that K is the number of arms, t is the current step and T is the total duration of the algorithm.

- ϵ-GREEDY: $\epsilon = \min(1, \sqrt{\frac{\log K}{t}})$
- FPL: $\eta_t = \min(1, \sqrt{\frac{\log K}{t}})$. Choose $\text{argmax}_i \hat{\mu}_t^i + Z_t^i$ where $\hat{\mu}_t^i$ is the empirical mean reward for ith arm by time t and $Z_t^i \sim \text{Exp}(\eta_t)$
- BOLTZMANN-GUMBEL: $\beta_t(i) = \sqrt{\frac{1}{N_t^i}}$ where N_t^i is the number of iterations where ith arm is chosen by time t. Choose $\text{argmax}_i \hat{\mu}_t^i + \beta Z_t^i$ where $Z_t^i \sim \text{Gumbel}(0, 1)$.
- EXP3: $\gamma_t = \min(1, \sqrt{\frac{K \log K}{(e-1)t}})$

Appendix B Simulation

B.1 ABM Calibration

With the primary goal of capturing the non-linear dynamics of trading activity and funding rate, we selected various indicators of trading activity by performing funding rate granger causality tests using 3 months of dydx exchange data on 3 perpetual futures markets BTC-USD, ETH-USD, and SOL-USD sampled at a 1-minute frequency. After conducting stationarity tests, we selected the following 6 indicators per interval: (1) total number of buys n_b, (2) total number of sells n_s, (3) order imbalance $n_b - n_s$, (4) cumulative amount of buys (in number of contracts) a_b, (5) cumulative amount of sells a_s, (6) order amount imbalance $a_b - a_s$.

Instead of calibrating by the method of simulated moments, we decided to calibrate directly to the empirical distributions of each indicator using the Kolmogorov–Smirnov (K-S) test. Our calibration objective function is to minimize the sum of the K-S statistics D_n from performing two sample K-S tests between the empirical distribution F_n^e and simulated distribution F_n^s of each indicator $i \in I$ averaged over the Monte Carlo dimension d:

$$\text{minimize} \quad \sum_{i \in I} \left(\frac{1}{d} \sum_{k=1}^{d} D_n^i \right)$$

where n is the number of samples.

Our model has 9 free parameters: number of agents, mean wealth, and time horizon upper bound for each agent type. However, by simulating and observing that funding rate arbitrageurs act like stabilizing agents in the context of derivatives, arbitraging away the difference in mark and index price, we reduced the number of free parameters from 9 to 6 by using the same set of parameters for both fundamentalists and arbitrageurs.

Next, we conducted one-factor-at-a-time sensitivity tests [4] to test the responsiveness of the objective function and verify the selection of indicators. Finally, we ran a particle swarm optimization algorithm to minimize our calibration objective with the following hyperparameters:

Swarm-size (S)	Inertia (w)	Cognitive coefficient (ϕ_p)	Social coefficient (ϕ_g)
47	-0.1832	0.5287	3.1913

as suggested by [8] and using the week of 09/10/2022-09/17/2022 from dydx ETH-USD perpetual futures market sampled at a 1-minute frequency (see Figure 3).

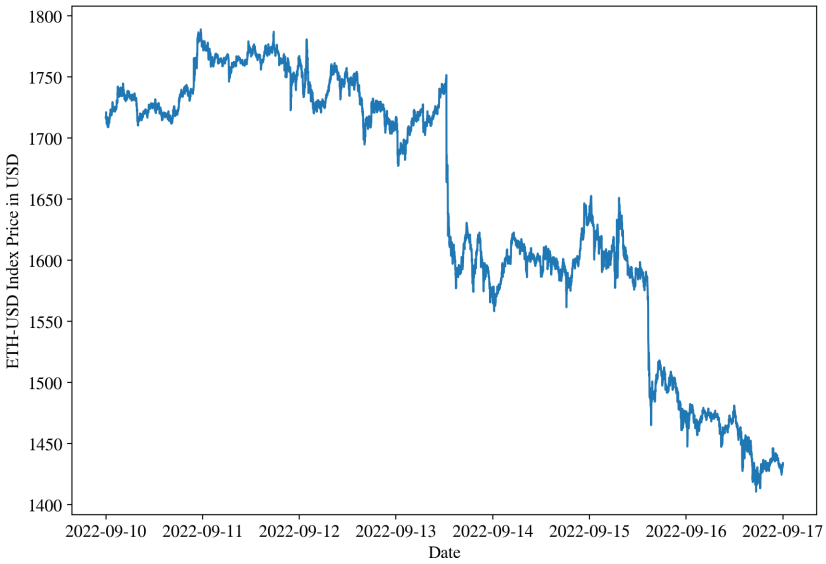

Figure 3: ETH-USD Index Price Path from dydx

Note that because of the expensive computational runtime of our ABM, we only conducted $d = 3$ Monte Carlo iterations of each free parameter combination, but found the K-S statistic quite robust *a posteriori*. Additionally, we only performed 25 iterations of the particle swarm optimization since we observed a slowly decreasing learning rate. Ultimately, we obtained the following free parameters:

Free parameter	Symbol	Value
Number of chartists	n^c	1383
Number of fundamentalists (arbitrageurs)	$n^f(n^a)$	361
Mean chartist wealth	w^c	188903
Mean fundamentalist (arbitrageur) wealth	$w^f(w^a)$	429081
Chartist time horizon upper bound	h^c_{max}	398 minutes
Fundamentalist (arbitrageur) time horizon upper bound	$h^f_{max}(h^a_{max})$	68 hours

Loosely speaking, since fundamentalists and arbitrageurs represent more informed traders with larger capital, we have $n^c > n^f(n^a)$ and $w^c > w^f(w^a)$ as expected. Using these free parameters to run our simulation, we obtained the following K-S statistics on a single Monte Carlo iteration:

Indicator	Symbol	K-S statistic
Number of buys	n_b	0.1715
Number of sells	n_s	0.1892
Order imbalance	$n_b - n_s$	0.0864
Cumulative amount of buys	a_b	0.1627
Cumulative amount of sells	a_s	0.1849
Order amount imbalance	$a_b - a_s$	0.0932

One limitation of these results is that all K-S test p-values were less than 0.05, indicating statistical difference between the empirical and simulated distributions of each indicator. This could be because agent behavior does not capture well enough trading activity or that the number of iterations of the particle swarm optimization algorithm was too small.

In addition to the free parameters of our model, we include several fixed parameters:

Fixed parameter	Symbol	Value
Chartist time horizon granularity	g^c	Minutes
Fundamentalist (arbitrageur) time horizon granularity	$g^f(g^a)$	Hours
Pareto shape	α	$\log_4 5$
Market friction upper bound	fr_{max}	100bps
Fee	f^t	1bps
Max expected funding rate	f^f_{max}	1bps
Max expected price change	ΔP_{max}	10%
Minimum order size	s_{min}	0.001

g^c was chosen as the lowest time interval in our simulation while $g^f(g^a)$ was chosen because funding was collected every hour on dydx. α was chosen to reflect the "80-20 rule" for the distribution of wealth amongst agents. fr_{max} and f^t were chosen based on popular crypto exchanges' fee structures. f^f_{max} and ΔP_{max} were chosen based on the selected week of trading data and s_{min} was chosen from the dydx ETH-USD perpetual futures market specification. c

B.2 Results

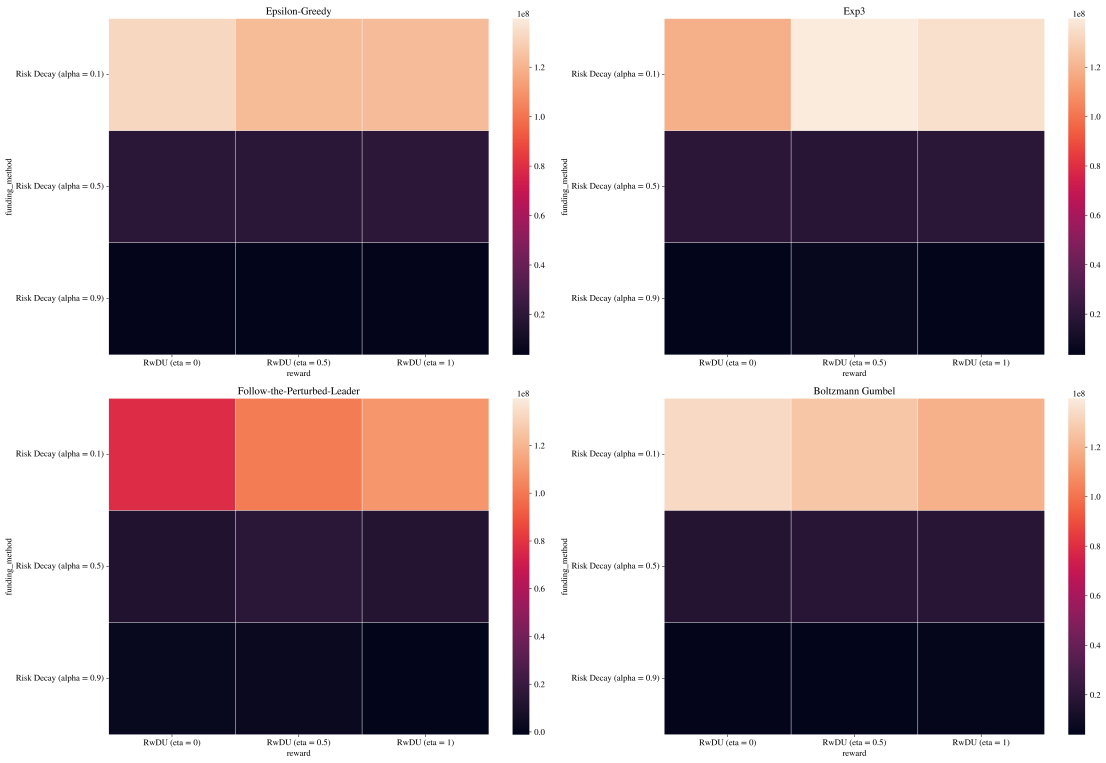

Figure 4: PnL Heatmap of different bandit algorithm

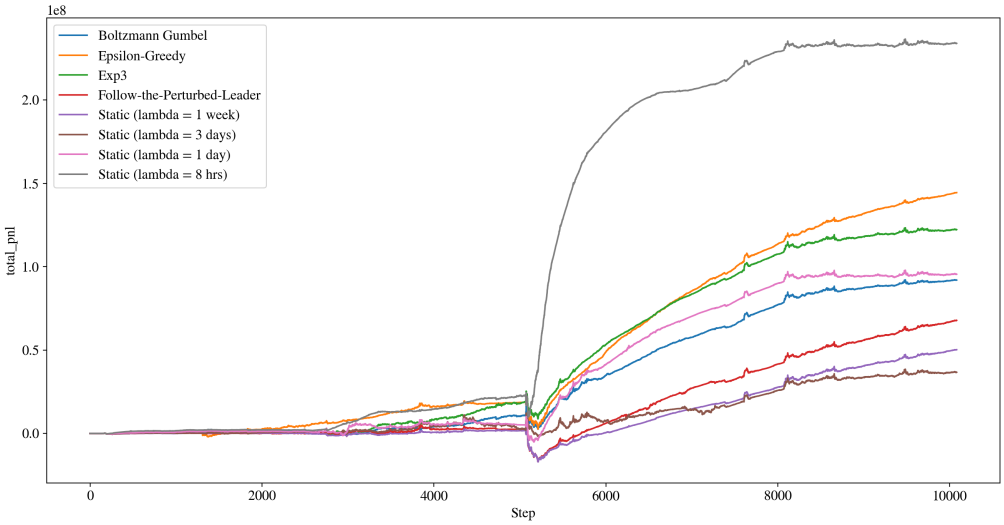

Figure 5: PnL under each strategy in a single run ($\alpha = 0.1$, $\eta = 0.5$)

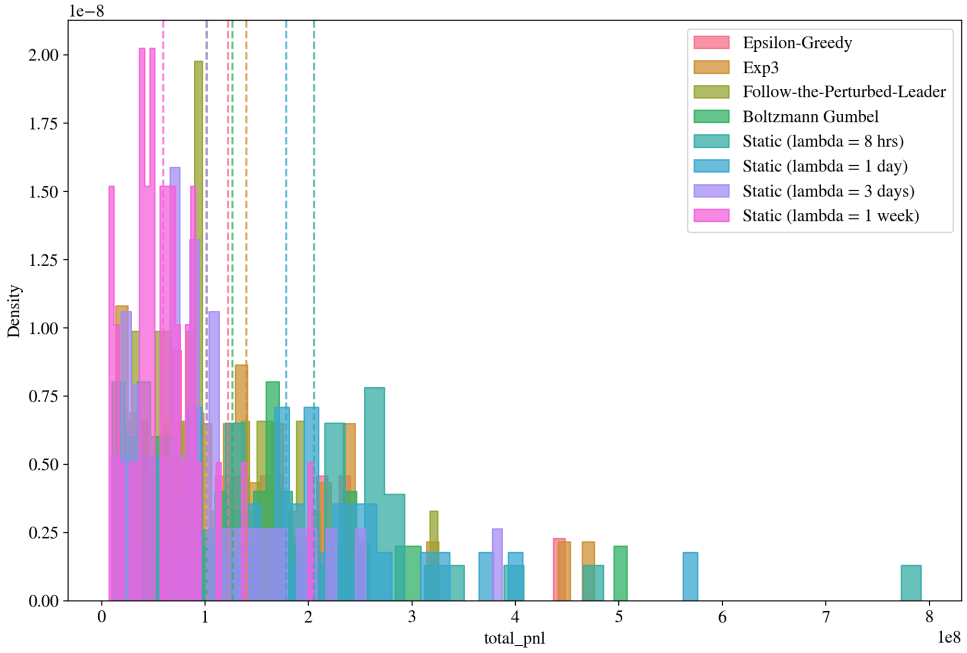

Figure 6: PnL distributions under each strategy ($\alpha = 0.1$, $\eta = 0.5$)