AERA

Risk-Aware DAO Treasury Management

October 2022

aera@gauntlet.network

Abstract

As decentralized finance (DeFi) continues its growth trajectory, peaking at over \$250 billion in assets locked in smart contracts, it is natural to ask what primitives are needed to reach a multi-trillion dollar market size. Improving capital efficiency within the decentralized realm is significantly more challenging than in traditional financial markets due, in part, to a lack of identity. One increasingly common way to improve capital efficiency in on-chain protocols is to have insurance funds that cover deemed shortfall events. Aave and Synthetix both rely on these stake-based insurance funds to ensure that trading or borrowing risky assets is safe for capital providers. However, in times of extreme duress, these treasury reserve funds themselves are likely to run into failure conditions as they have to cover their liabilities in a depreciating asset (*e.g.* the native governance token). In this paper, we present Aera, a mechanism for providing decentralized autonomous, risk-aware treasury management to decentralized treasury reserve funds. This mechanism allows treasury reserve funds to automatically decorrelate assets used to pay insurance liabilities from the protocol's native token, akin but not equivalent to reinsurance. We utilize theoretical and numerical arguments to justify that this mechanism achieves better capital efficiency and safety. We conclude by arguing that the mechanism used by Aera can be generalized to a two-sided prediction market for continuous governance of risk and incentive parameters.

1 Background

As decentralized finance (DeFi) grows on Ethereum and other smart contract platforms, there has been a dramatic increase in the usage of leverage in on-chain products. Aave, Compound, and MakerDAO, which are trustless, peer-to-peer lending contracts implemented as Ethereum smart contracts, collectively held over $20 billion in assets in July 2022. However, these protocols are *capital inefficient* when compared to traditional banks due to a lack of persistent user identities (or credit) and insurance. Intuitively, capital efficiency refers to the ability of a financial entity (be it a protocol or a bank) to generate returns on capital. More formally, capital efficiency can be described as the maximal ratio of the amount of revenue generating capital to the quantity of capital reserves needed to facilitate risky operations. DeFi's main benefits of transparency and pseudonymity inherently limit shrinking the denominator of this ratio, as more capital is needed to account for the inherent excess risk of allowing arbitrary borrowers in these systems. The goal of this paper is to argue that even in a transparent pseudonymous world, one can recover traditional capital efficiency with an active market where participants are incentivized to optimize capital efficiency.

Much of the innovation in DeFi since 2020 has involved the creation of new mechanisms to improve capital efficiency. One route for capital efficiency improvement came in the form of utilization of newer smart contract platforms such as Solana and Avalanche. In these settings, it was initially believed that higher throughput and lower latency implied that protocols could have more trust in the pricing mechanisms for loans (*e.g.* order books and/or oracles). Practically speaking, however, these mechanisms still had trouble within these environments [77, 31] and there were a number of incidents related to censorship and counterparty risk where capital efficiency had in fact worsened [102]. Moreover, improved automated market makers like Uniswap V3 [3] were often able to beat order books (while being on slower and more expensive networks) in capital efficiency terms [64, 130].

Before understanding how capital efficiency can be improved for protocols, we will first take a step back and analyze the landscape of risk products. In traditional finance, risk products that allow parties to transfer particular risks between one another in exchange for premia, are the main tools for improving capital efficiency. The main non-speculative demand for these products in traditional finance tends to be institutions who have long-term institutional objectives, while speculative demand ensures that there is ample liquidity for institutional trades [66]. While many risk products exist within DeFi, they have little uptake, especially amongst crypto-native institutions such as decentralized autonomous organizations (DAOs). We will argue that the structure of DAOs makes them unable to optimize their own asset composition and take advantage of risk products, leading to the low usage we see in practice. In particular, we posit that the lack of DAO participation in the risk product market is the main driver of capital inefficiency in DeFi.

As a solution to this problem, we will argue that DAOs need to incentivize their members to allocate assets to risk products via methods that do not require using on-chain governance. These methods will require DAO participants who aim to change asset compositions and allocations to risk products to put their own capital at risk (i.e., to have skin-in-the-game) when changing DAO parameters. If these participants improve a DAO's welfare (*i.e.* institutional objectives), then they should be rewarded with fees, whereas if they worsen the DAO's welfare they will face a financial penalty (*e.g.* slashing). We then present Aera as a protocol that aims to solve this problem through incentives to maximize DAO treasury reserve fund welfare. Finally, we will conclude by describing how capital efficiency for DAOs that hold risk in their insurance funds (such as Aave) demonstrates how these incentives could work in practice.

1.1 Allocation Problems in Traditional and Decentralized Finance

Many of the allocation problems faced by DAOs resemble allocation problems faced by institutions and non-profit organizations. In this section, we will first provide a brief background of how treasury management in the traditional world is handled. Then we will consider some case studies of how large DAOs have unsuccessfully been able to execute allocation strategies. We posit that this is partially due to the heavily adversarial nature of allocations within DeFi. However, by formulating the allocation problem as an optimization problem, we argue that it is possible for a DAO to utilize arbitrage mechanisms rather than governance to optimize its treasury allocation.

Risk Products and Institutional Flows. There exist a plethora of risk products (which are usually derivatives or structured products) to allow DeFi users and traders to hedge risk and utilize less collateral. These include, but are not limited to, credit default swaps (*e.g.* Saffron [114], Opium [17], Anzen [12], Gro [58]), options protocols (*e.g.* Opyn [49, 50], Panoptic [76], Primitive [125]), exotic options (*e.g.* Cega [115]) and portfolio insurance (*e.g.* Risk Harbor [117]). While these products have garnered some usage, the majority of them have had trouble attracting volume greater than single digit basis points of the notional value of the exposures that they cover [2, 129]. One of the main reasons for the lack of adoption of these products relative to their traditional finance counterparts is the lack of institutional flows. In traditional markets, institutional demand for derivatives and hedging products ensures that there is a minimum percentage of derivative consumption that is driven by non-speculative demand [48, 66]. For instance, airlines hedging oil and energy costs in the futures markets or large asset managers hedging industry-wide portfolios with credit default swaps are examples of institutional flow. In cryptocurrencies, however, there is little institutional demand for on-chain derivatives outside of DeFi options vaults (DOVs) such as Ribbon Finance [78, 56].

One of the main issues with institutional demand for derivatives in the cryptocurrency world is that the only credible on-chain institutions that market are DAOs. DAOs generally utilize token-based or NFT-based governance to allow participants to vote on programmatic code changes to everything from risk parameters to how DAO treasury assets should be spent. However, DAO governance tends to be slow and reactionary versus nimble and proactive [30]. Since holders of risk products tend to have higher-frequency trading behavior (especially around rare market events), it is generally hard for a DAO to purchase risk products directly. In particular, DAO governance is often a slow, methodical process which makes make high-velocity decisions challenging. Asking a DAO to manage a portfolio of derivatives would be akin to asking the US Congress (which passes and enacts at most 1230 laws per year [57]) to market make derivatives (which requires billions of transactions per year [81, 45]) by passing laws to execute particular trades. This inefficiency of DAO purchasing is reflected in the lack of uptake of a number of DAO-focused derivative products such as UMA's KPI options [27] or Element's fixed-interest treasury management [87].

We posit that the main issue in uptake of these products isn't that the products don't fit DAO needs, *per se*. Instead, the incentives to perform optimal treasury allocations to such fixed-income, options, and spot products are misaligned and such treasury diversifications happen too infrequently. Unlike companies, DAOs tend to make a small number of very high impact decisions as governance votes involving financial products take an enormous amount of education and detailed research to execute successfully. As such, actions that require high-frequency decision making (such as portfolio rebalancing) are difficult for DAOs to do directly. This 'time-scale separation' between governance and portfolio rebalancing is fundamental to why treasury diversification for capital efficiency in DeFi has been stunted so far. Before discussing solutions to this problem, we will first illustrate this point by studying the allocation strategies of DAOs that have had hundreds of millions of dollars of assets in their treasury.

DAO Allocation Case Studies. As a case study of this, consider the SushiSwap treasury diversification votes of 2021 and 2022 [1]. DAOs, which generally have large treasuries predominantly denominated in their own governance token, often have to sell their own governance token to fund operations (*e.g.* development and marketing costs, security auditing, user incentives, and risk management). However, tokenholders are generally averse to such sales as they lower the value of their tokens (*e.g.* the DAO is selling a large block of governance tokens, causing the price to go down). Moreover, there is adverse selection for DAOs as they explicitly tell the whole market that they are selling at a particular time, allowing others to front-run them. Given these adversarial incentives, even DAOs like SushiSwap, MakerDAO, and Aave which have/had treasuries with hundreds of millions of dollars of assets, are unable to perform more than one or two diversifications a year if at all. SushiSwap would have had roughly \$46.9M more in capital had it's diversification proposal succeeded.[1] In Aave's case it has been difficult to get participants to align on the mechanism for diversification and its goals [19]. Finally, MakerDAO had the controversial LOVE-001 vote which aimed to fund various core teams, which inevitably failed [69].

Decentralized Allocation Problems. Examples strongly suggest that DAOs are inefficient vehicles for risk capital allocation. This is similar to traditional public institutions such as endowments and non-profit foundations that are only able to make a small number of allocation decisions per year. Such institutions usually allocate their funds to more nimble managers, such as hedge funds or venture capital funds. These fund managers have incentives (*e.g.* management and performance fees) that are correlated to how well they meet

[1]The DAO had initially intended to sell \$60,000,000 worth of SUSHI when SUSHI/USD was \$6.05; the current price as of this writing is roughly \$1.32 [1]

institutional objectives. These other funds can deploy and reallocate capital more efficiently to different asset classes in order to maximize institutional objectives. Note that even in traditional finance, institutional objectives are generally not strictly profit-maximizing, *cf.* ESG investing [46, 53].

In the traditional, centralized world, adherence to institutional objectives is enforced via legal contracts and recourse. This, however, is not possible for DAOs, as they need to be able to withdraw their funds at will. For instance, if a DAO's treasury has to serve as an insurance fund, the DAO needs to be able to withdraw funds from an external fund manager at arbitrary times (*i.e.* whenever a shortfall event is realized). Moreover, DAOs need to minimize the number of interactions that they have with any on-chain function call, as each new conditional interaction will require a new governance proposal. This intimates that two necessary qualities that a decentralized allocation protocol needs are at-will withdrawals from on-chain fund vehicles and a bound of at most two interactions needed (*e.g.* deposit and withdrawal).

Historically, DAOs have deployed their treasuries to other fund-like on-chain vehicles, such as Yearn Finance or Lido stETH vaults [90, 132]. However, these treasury deployments have not been connected to the capital and risk requirements of the DAO (*e.g.* a DAO's analogue of institutional objectives). For instance, a DAO whose treasury is used as an insurance fund for a DeFi protocol needs to ensure that the assets in the treasury are worth more than the liabilities held by the protocol. But a simple deposit of treasury assets into on-chain yield vehicles (such as Yearn) is not guaranteed whatsoever to be correlated with protocol liabilities.

Another example of this is a 'runway' metric: if the DAO has expenses denominated in stablecoins but wants to ensure that it has enough runway to pay for future development costs, it needs to ensure that its stablecoin-denominated assets are always worth more than the desired runway. As described before, however, this goal can be contentious as DAO tokenholders are averse to selling assets to rebalance the treasury since it will likely lower the value of their assets. Moreover, DAOs will need to constantly be updating allocations to native token denominated yield versus stablecoin yield to maximize runway — something that is not accounted for within Yearn-style vaults.

How can we ameliorate this problem? Since there is no legal recourse in this setting, we will need to construct an incentive mechanism that encourages DAO tokenholders, analysts, and speculators to assist DAOs with the continuous adjustment of asset composition to achieve its institutional objectives. Aligning the incentives of all three parties is crucial to ensuring that such a system can solve the decentralized allocation problem and be a positive outcome for all stakeholders in a DAO. The remainder of this paper will focus on mechanisms for aligning incentives amongst these parties.

Decentralized Allocation is an Adversarial Optimization Problem. Traditionally, most asset allocation problems can be viewed via modern portfolio theory, which utilizes convex optimization to provide portfolio recommendations [84, 22, 73]. In this framework, an asset allocator has an initial wealth W, a portfolio π that allocates W to a universe of

N_{assets} assets. Additionally, they have a convex objective function f that maps a portfolio π and an on-chain environment Σ to some measurement of asset quality. The portfolio represents the relative allocation to different assets (*e.g.* 40% to stablecoins, 60% to ETH) whereas the environment represents the current on-chain state of the protocol. For instance, the outstanding loan book of a lending protocol serves as on-chain state that is meant to be hedged by the protocol treasury, like the Aave safety module. On-chain state allows us to characterize treasury needs such as specific liabilities a protocol could have in the future. In the case of lending protocols, these could be potential short falls derived from on-chain state of the loan book.

Including the environment in the optimization problem is crucial, as certain portfolios work best under certain market conditions (which are represented by as the environment[2] at time t, Σ_t). For instance, $f(\pi, \Sigma_t)$ could measure how diversified a portfolio is or measure the expected risk-adjusted returns at time t. However, risk-adjusted returns in a bear market are quite different from risk-adjusted returns in a bull market and any treasury allocation strategy needs to account for this. In traditional finance, a centralized allocator constructs a new portfolio π^* by taking in new market information — updating the environment as $\Sigma_t \leftarrow \Sigma_{t-1} \cup \{\text{new information}\}$ — and solving an optimization problem $\pi^*(\Sigma_t) = \sup_\pi f(\pi, \Sigma_t)$. Unfortunately, this formulation of the allocation problem does not translate cleanly to the decentralized environment.

Firstly, a DAO needs to take in information and treasury allocations from n parameter submitters (*i.e.* DAO members who provide recommendations for treasury allocations), each of whom have a personal objective function $f_1, \ldots, f_n$ that can be dramatically different from the DAO's objective function f_{DAO}. It is possible (and very likely!) that the optimum values of the DAO's objective function, $\mathsf{OPT}(f_{\text{DAO}}) = \sup_\pi f_{\text{DAO}}(\pi, \Sigma_t)$, does not intersect with the the optimum values of any of the participants.[3] Concretely, suppose that participant i wants to buy the DAO's governance token at a cheap price so they submit a portfolio to the DAO that says the DAO's portfolio should be 100% in stablecoins and 0% in the DAO governance token. This way, if the DAO rebalances to this portfolio, then it will sell all of its DAO governance tokens, making the price cheaper for the participant to purchase more tokens. However, the DAO's optimal portfolio might be 50% governance tokens and 50% stablecoins and not the participants' submitted portfolio. Formally, if the portfolio tendered is $\pi = (\%\text{ stablecoin}, \%\text{ governance token})$, this could be written as $(0.5, 0, 5) \in \mathsf{OPT}(f_{\text{DAO}})$ but $(1.0, 0) \not\in \mathsf{OPT}(f_{\text{DAO}})$ and $(0.5, 0, 5) \not\in \mathsf{OPT}(f_i)$ but $(1.0, 0) \in \mathsf{OPT}(f_i)$. As such, we need to assume that portfolios submitted from participants will be adversarial[4] towards the DAO

[2]Formally, one should think of an environment as a filtration $\{\Sigma_t\}_{t\geq 0}$ on a probability space defined by the set of allowable on-chain transactions at time t and the historical transactions at times $s < t$

[3]More formally, one can formalize the adversarial optimal assumption as follows: For fixed $\epsilon > 0$, let $B(x, \epsilon) = \{x' : \|x' - x\| < \epsilon\}$ be an ϵ-neighborhoood of $x \in \mathbf{dom}\, f_{\text{DAO}}$. Then we have adversarial participants if $\Pr[B(x', \epsilon) \cap B(X, \epsilon) = \varnothing : x' \in \cup_i \mathbf{dom}(f_i), x \in \mathbf{dom}(f_{\text{DAO}})] = g(\epsilon)$ for a 'slowly' non-increasing positive function $g : \mathbf{R}_+ \to [0, 1]$. We note that the randomness comes from the choice of $f_1, \ldots, f_n$, which is unknown to f_{DAO}.

[4]Technically, this includes a more generate set of actors who are simply unaligned or misaligned rather than purely adversarial. For instance, you can have cost functions f_i whose optima do not coincide with those of f but which aren't exactly "adversarial". Consider a two player game $f_1(x, y) = x^2 + (y - 1)^2$ and

(*e.g.* do not have the same objective functions and/or mimima).

Secondly, DAOs are relatively restricted in how they can execute portfolio rebalances. They need to use on-chain mechanisms such as trading with automated market makers, running auctions (such as Gnosis Auction [55]), or using a lending protocol. The precise execution model used can interact with how truthful participants' tendered portfolios are, however. Suppose again that participant i desires to buy governance tokens at a lower price than the current market price. First, suppose that the DAO uses an automated market maker consisting of the governance token and a stablecoin for execution. If participant i is both an arbitrageur of the automated market maker (*e.g.* trades between the AMM and an external market) as well as a parameter submitter, participant i's incentive is to always tender portfolios that force the DAO to make large asset allocation changes. This can lead to drainage of the DAO treasury via arbitrage losses (which are profits for participant i). This example illustrates that portfolio submitters must have 'skin-in-the-game' and have to take losses (measured relative to how much they decrease f_{DAO}) if they cause the DAO to incur losses.

Finally, DAOs need to ensure that portfolio submitters are incentivized to regularly and effectively submit portfolios. If the DAO does not provide an incentive, such as a management or performance fee, to submitters, then one cannot expect submission quality to be high enough that f_{DAO} is often near an optimum value. To see why, let us first view the portfolio allocation problem as form of online optimization — at each round $t = 1, 2, 3, \ldots$, each portfolio submitter submits a portfolio $\pi_i(t)$. The DAO then aggregates these into a final portfolio $\tilde{\pi}(t)$ and then executes trades to enter into this portfolio. If $f_{\mathrm{DAO}}(\tilde{\pi}(t), \Sigma_t) > f_{\mathrm{DAO}}(\tilde{\pi}(t-1), \Sigma_{t-1})$ and the ith submitter's submission pushed[5] $\tilde{\pi}(t-1)$ towards $\tilde{\pi}(t)$, then they are rewarded. Without adding in a reward, one should expect that $\pi_i(t)$ is sampled uniformly at random and our rate of convergence towards maximiziers of f_{DAO} will be slow. Moreover, if continued 'good' submissions (*e.g.* submissions that improve the objective value f_{DAO}) are not rewarded, then submitters will not be incentivized to spend the effort to find the best optima. Incentivizing portfolio submitters with fees that reflect their participation and quality in the network will ensure that they try to find portfolios that incentivize $f_{\mathrm{DAO}}(\tilde{\pi}(t), \Sigma_t) > f_{\mathrm{DAO}}(\tilde{\pi}(t-1), \Sigma_{t-1})$, which is crucial for the DAO being able to finding optimal allocations faster.[6]

These example provide the three pillars of the what all functional decentralized allocation protocols need to assume:

1. **All Actors are Adversarial**: Decentralized allocation algorithms are able to make progress in spite of having to assume all submissions are adversarial (akin to adversarial online convex optimization [5, 119])

$f_2(x, y) = y^2$. Then, player 2 doesn't care what player 1 does, and player 1's optimizer is at $(x, y) = (0, 1)$ whereas player 2's is at $y = 0$.

[5] This is not uniquely defined and will vary for different applications; however, intuitively, one can consider $\mathsf{corr}(\pi_i(t) - \tilde{\pi}(t-1), \tilde{\pi}(t) - \tilde{\pi}(t-1)) > 0$ for a correlation metric corr

[6] In our follow-on paper, we will formalize this by showing that fees increase the rate of convergence of f_{DAO} to an optimum. A similar result was shown for convex objectives in [100]

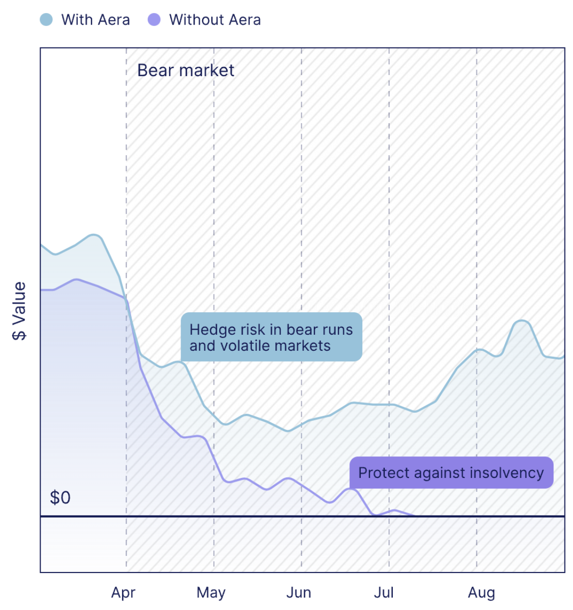

Figure 1: Stylized figure of how Aera would behave in a bear market as a hedging vehicle for DAOs

2. **Different types of actors can collude to worsen portfolio performance**: Ensuring that collusion is at best a zero-sum game (*e.g.* submitters lose if arbitrageurs are able to profit) is crucial to ensuring that an adversary cannot drain the DAO treasury

3. **Submitters need to be incentivized to speed up the distributed optimization algorithm**: By paying submitters with fees proportional to reputation and/or contribution to objective function improvements, one ensures that submitters do not randomly search the space of portfolios and have a better rate of convergence than random search

In follow-up work, we will prove that under some mild assumptions about price time-series, particular configurations of Aera are able to make progress towards an optimum of f_{DAO} under these three assumptions.

1.2 Aera: Decentralized Allocation as a Protocol

The past section has distilled the DAO treasury management problem into a decentralized, adversarial, incentive-driven online optimization problem. A natural question to ask at this juncture is: "Can we use ideas from decentralized finance and adversarial online optimization to provide practical solutions to this problem?" To answer this question, we combine tools from automated market making, model selection in machine learning, and algorithmic game theory. Figure 1 provides a stylized view of how an Aera vault would ameliorate DAO treasury allocation in a bear market.

We will first introduce *Aera* as a protocol that constructs a competition amongst agents who submit DAO treasury allocations. Figure 2 shows how the different agents and the DAO interact with the treasury. Agents, known as *vault guardians*, participate in an iterated skills-based competition with the goal of maximizing their individual rewards paid out from the DAO for participating. At each round of the game, vault guardians tender a portfolio for a DAO's treasury. In order to participate, vault guardians will have to stake assets that can be slashed if their allocation causes the DAO to suffer a loss. Arbitrageurs who spot price discrepancies between a DAO treasury's implied portfolio value and the rest of the market will also able to participate in the competition by proposing improved DAO treasury allocation. This allows the DAO to 'outsource' slashing, as arbitrageur gains correspond to allocator losses. In particular, arbitrage losses that the DAO treasury faces are passed on via slashing to the allocator if the losses are sufficiently large. This ensures that even if arbitrageurs and vault guardians collude, their maximal profit is bounded and the game between these two agents and the DAO is zero-sum.

One key feature of Aera is that it forces *time-scale separation* between the different participants. The depositing DAO is a depositor with very long time preference who primarily conducts two actions: depositing and withdrawing. The DAO also authorizes a specific algorithm for selecting optimal DAO treasury allocations among Guardian submissions. They do not have to make any predictions about the assets in the vault, but they do have to pay fees to participate. Vault Guardians, on the other hand, have to theorize on optimal DAO treasury allocations. There is a minimum time period E for which they must submit a portfolio and they are graded on how their portfolio performs over the time $t, t+1, \dots, t+E$ (see §2 for a concrete example). This forces the vault guardian, who stakes assets to suggest an optimal allocation, to have non-trivial predictive power about the DAO's objective function value and the external market, should they want to be profitable. Finally, arbitrageurs are allowed to interact with the vault's portfolio and trade against it at any time (*e.g.* when they spot an arbitrage between the DAO's quoted price and an external market).

Allocations and Aggregation. Each DAO treasury generically has a universe U of assets that it is willing to invest in. The set U of allowable assets is a relatively static choice and should be updated infrequently via on-chain governance. For instance, U could include ERC-20 representations of assets such as stablecoins, Opyn options, Element fixed-interest loans, or automated market maker LP shares. An allocation is represented by a portfolio $\pi \in \{p \in \mathbf{R}_+^{|U|} : \mathbf{1}^T p = 1\}$. The portfolio π represents the percentage of the treasury that is allocated to each of these assets.

Once vault guardians submit their portfolio allocations $\pi_1, \dots, \pi_n$ to the protocol, an aggregation rule Agg is used to construct a single portfolio that the DAO then executes trades to enter. The aggregated portfolio, $\tilde{\pi} = \mathsf{Agg}(\pi_1, \dots, \pi_n)$, can be thought of as a multidimensional generalization of aggregation rules used in price oracles, such as Chainlink's medianizer [23] or Pyth's confidence interval weighted aggregation rule [13]. We discuss aggregation rules and methodology in §3.

Upon receiving the aggregated portfolio $\tilde{\pi}$, the Aera protocol takes treasury assets and

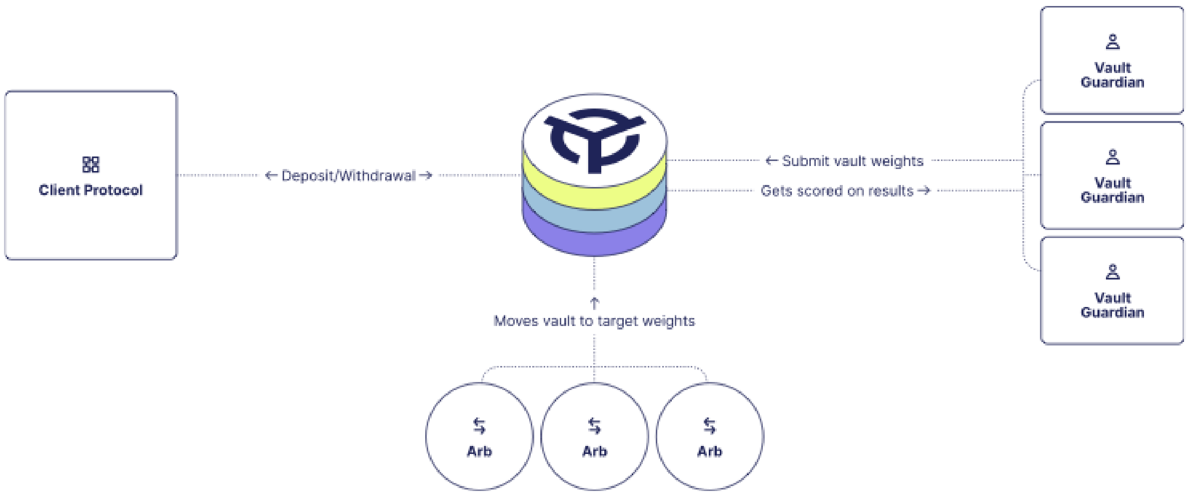

Figure 2: Pictorial description of the different agents involved within Aera

puts them in a particular constant function market maker (CFMM) [9, 8] that is adapted to ensure that the relative proportions of the assets are equal to $\tilde{\pi}$. In this way, the DAO will be a liquidity provider to a CFMM that helps execute the trades for adjusting the portfolio to $\tilde{\pi}$. As a CFMM automatically quotes a price for every pair of assets within the share, $p_{i,j}, i, j \in U$, arbitrageurs can trade against the portfolio until the relative asset proportions are equal to $\tilde{\pi}$ [39]. The portfolio weights will be at $\tilde{\pi}$ when the prices quoted by the market maker are in equilibrium with external prices (*e.g.* at a centralized exchange).

Recall that CFMMs impair a loss upon liquidity providers (known as impermanent or divergence loss [10]) due to the concavity of CFMM trading functions. If allocators collectively submit portfolios where $f_{\text{DAO}}(\tilde{\pi}, \Sigma_t)$ deviates significantly from $f_{\text{DAO}}(\pi^*(\Sigma_t), \Sigma_t)$, then they increase the arbitrage loss felt by liquidity providers (see Appendix A). This loss can be computed on-chain and passed through to the allocated by slashing their staked assets. As such, arbitrageurs serve as an important feedback mechanism for ensuring that the allocators have skin-in-the-game for their predictions.

Prior work on Aggregation. Choosing an aggregation rule is a well-studied problem in machine learning and algorithmic game theory. Recent work [100] has shown that there exist optimal aggregation rules for convex loss functions and one can directly compute fees for how much to pay submitters for predictions in such a scenario. Other work, especially in the field of federated learning, has focused on adversarial resistant aggregation rules for gradient descent [18], data poisoning attacks [40], differential privacy [20], and decentralized neural network training [89]. However, the majority of these aggregation rules are too computationally expensive for a blockchain environment. The mechanism of [100] involves computing a Fenchel conjugate of a loss function, while the mechanisms from federated learning generally use trimmed means (*e.g.* compute the mean after throwing out particular quantiles of received predictions).

Recent work on robust aggregation [110] argues that weighted, geometric medians are sufficient for robust aggregation. Aera will use this class of aggregation rules as they are significantly simpler to compute and can be evaluated in on-chain environments. We note that Pyth [13] also uses a variant of a weighted median aggregation rule. Moreover, the weights in these aggregation rules naturally correspond to the reputation scores for each submitter so that higher reputation submitters are able to have a larger impact on the final aggregation value. This, coupled with the proper incentives for providing good allocations, can be used to ensure that we are close to $f_{\text{DAO}}(\pi^*(\Sigma_t), \Sigma_t)$ (which will be formally shown in a follow-on paper). If there is excess veriable compute capacity in the future (*e.g.* on a Zero-Knowledge rollup such as Starkware [124], Polygon [97] or Scroll [121]), Aera can migrate to a higher accuracy aggregation rule.

Grading and Incentivizing Allocators. A key component of the Aera protocol is that a participant's allocation submission is *graded* based on their contribution to how much they changed f_{DAO}. If a submitter deviates wildly from rational, optimizing submitters' values and this causes the final portfolio to move away from the optimal value, they should receive less in fees or even a slashing. Deciding how to attribute how much a participant's submission changes an aggregate output is a difficult problem but has been extensively studied in federated and collaborative learning. We will adapt ideas from these areas to utilize when grading Aera submitters.

As a concrete example, suppose that a participant submits a portfolio $(0.1, 0.2, 0.7)$ whereas the optimal portfolio is $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) \in \mathsf{OPT}(f_{\text{DAO}})$. The loss to the DAO, L, should the aggregate submission have been $(0.1, 0.2, 0.7)$, is $L = f_{\text{DAO}}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}) - f_{\text{DAO}}(0.1, 0.2, 0.7)$. Since f_{DAO} (or simplified approximation of it) can be evaluated using only on-chain state, a smart contract can compute L and penalize the agent if L is bigger than some threshold, *e.g.* penalize if we are 10% away from the optimum $L > 0.1 \times f_{\text{DAO}}(0.1, 0.2, 0.7)$ or if the loss is greater than 10% of the total treasury in ETH terms. The choice of such thresholds can dramatically impact the performance of a collaborative optimization problem (see, *e.g.* [18] and [89]). Choosing these thresholds for when a submitter is penalized depends on properties of the assets involved, including their volatility profiles and conditional value-at-risk.

Moreover, just as other collaborative learning problems that have Schelling or focal points, we also have to reward agents who submit allocations that concentrate around a particular value. The statistical justification for this is that concentrated predictions provide tighter confidence intervals that can be used to evaluate prediction quality. While focal points have a number of known collusion issues [52], providing tighter confidence intervals is important in multi-round, iterated games as it is the main way participants reveal information about their generative processes that provide predictions [70, Ch. 3]. As such our grading methodology for submitters rewards submitters separately for providing a parameter that improves the objective function and for providing confidence in the system. We note that our grading methodology differs significantly from Pyth [13], as we end up grading performance against f_{DAO} as opposed to grading based on tightness of quoted confidence intervals.

We subsequently use the grades of each submitter to construct relative rankings in order to determine what percentage of fees a particular submitter will get. Recall that the Shapley value of an n-person coalition game with value function v describes the contribution that the ith player contributed towards an equilibrium [120]. In our scenario, if the grades for each submitter represent the value provided by the ith player, then the Shapley value represents how much player i contributed to the final portfolio. For this distribution of fees, we measure the impact of the ith submitter using a weighted Shapley value constructed from the individual grades [71, 120]. Relaxations of weighted Shapley Values that are efficient to compute have been utilized in machine learning for model attribution and selection for both decision trees and neural networks [85, 127].

As we can view each submitter as submitting a 'model' (their weights for the treasury portfolio) and the aggregation rule as constructing an ensemble, we can use the Shapley values to measure submitter performance. In turn, this performance is used to distribute a fixed set of fees based on players' relative performances. If the total fees to be spent by the DAO on block height h are $\gamma(h)$ and Λ_i is player i's Shapley value, player i earns $\gamma(h)_i$:

$$\gamma(h)_i = \gamma(h)s_i(h)\left(\frac{\max(\Lambda_i, 0)}{\sum_i \max(\Lambda_i, 0)} + \frac{\min(\Lambda_i, 0)}{\sum_i |\min(\Lambda_i, 0)|}\right)$$

where $s_i(h)$ is the stake of player i at height h. This says that a player who had a positive contribution to improving f_{DAO} gets pro-rata fees relative to the set of all players with positive contribution ($\Lambda_i > 0$). In practice, the precise calculation of this value is approximated and there are a number of other practical desiderata regarding fee distribution (and slashing), which are covered in Appendix D.

1.3 Comparison to Prediction Markets and Futarchy

There are a number of similarities between Aera and futarchy [61, 24], a mechanism for executing governance decisions via prediction markets. First we claim that they both require participants (who are submitters and arbitrageurs in Aera) to make non-trivial calculations that take advantage of asymmetric information. Secondly, they both use automated market makers to ensure that liquidity is present for participants to make informed bets. However, the three main differences that ensure that Aera is able to avoid some of the pitfalls of futarchy (which has not been successful in practice) are:

1. **Usage of CFMMs.** The types of market makers used for futarchy, such as the logarithmic market scoring rule, have been unsuccessful at attracting usage outside of niche election markets. CFMMs, however, have processed trillions of dollars of transactions and are easy to setup (see below for details on the computational complexity of logarithmic market scoring rules)

2. **Time-Scale Separation of Market Participants.** Submitters calculate and theorize outcomes and lock up capital for a relatively long period of time whereas arbitrageurs (who in essence execute the portfolio rebalance for the DAO) only have to lock up capital for short periods of time

3. **Predictions are only of on-chain state.** Aera facilitates predictions of on-chain state only (*e.g.* size of treasury in the future, expected loan book size) versus exogenous, off-chain events. This makes Aera a closed-loop system in that the objective function, grading, and incentive distribution can be executed on-chain provided there is data availability.

We will discuss these three differences in detail after describing the similarities between Aera and futarchy.

Similarities with Futarchy. We first argue that successful submitters necessarily need to have non-zero predictive power to continue to be successful in Aera. This is equivalent to the futarchy requirement that some agents have non-trivial asymmetric information in order to reach a Bayes rational expectations equilibrium [61, 108]. First, note that submitters are optimizing the objective function f_{DAO}, which is a function only of the current environment Σ_t (as opposed to having path dependence, *i.e.* dependence on Σ_s, for $s < t$). We claim that to observe if a portfolio π is actually improving f_{DAO}, we need to observe the objective function after some period to compute the impact of rebalancing to π. In other words, we need measure $f_{\mathrm{DAO}}(\pi, \Sigma_t)$ and $f_{\mathrm{DAO}}(\pi, \Sigma_{t+\delta})$ for some $\delta > 0$.

Suppose for a contradiction that this were not true. Then we could find an element of $\mathsf{OPT}(f_{\mathrm{DAO}}(\pi, \Sigma_t))$ via a deterministic algorithm such as gradient descent or an interior point method that we could evaluate on-chain. However, such a deterministic approach is vulnerable to being front-run as DAO asset purchases would become highly predictable (*e.g.* would be the target of maximal extractable value searchers [75]). This would lead to $f_{\mathrm{DAO}}(\pi, \Sigma_{t+1})$ being worse than $f_{\mathrm{DAO}}(\pi, \Sigma_t)$, in spite of choosing an optimum portfolio. Moreover, the longer the observation window δ, the more confidence we can have that $f_{\mathrm{DAO}}(\pi, \Sigma_{t+\delta})$ is better or worse than $f_{\mathrm{DAO}}(\pi, \Sigma_t)$. This comes directly from the fact that Σ_t is a filtration on a probability space generated by all historical on-chain transactions and valid sequences of on-chain transactions that can be executed at block height t.

Combined, these demonstrate that a submitter has to have predictive power about $\Sigma_{t+\delta}$ in order to improve $f_{\mathrm{DAO}}(\pi, \Sigma_t)$ with positive expectation. This can only happen if the agent has non-trivial information about future prices and on-chain state (*e.g.* loan book) behavior. Secondly, most futarchy proposals (which were formalized by Pennock, et. al [108]) use automated market makers to ensure that players with excess information can bet on a future state. In Aera, the protocol commits to an aggregated portfolio held in an automated market maker for the length of an epoch δ. Arbitrageurs trade against that portfolio and effectively 'grade' how well the aggregate portfolio does over an epoch.[7] If submitters' aggregated information predicts the correct portfolio to maximize f_{DAO}, then the value of the portfolio at time $t+\delta$ should be maximized via arbitrageurs and informed trades [10]. We can view the arbitrageurs' trades against the portfolio as a process to guide the portfolio to $\Sigma_{t+\delta}$, where submitters win if they predicted information about that state correctly and lose otherwise.

[7]Arbitrage losses are only one component of the grading process. For instance, for the risk-aware treasury management described in §2, the parameters chosen by the submitters also need to cover the liabilities of the depositing DAO in order for them to receive rewards.

Market Structure Differences with Futarchy. Futarchy seeks to override an existing status quo with a completely new market that needs to be created from scratch. On the other hand, the status quo for DAOs is ineffective and Aera is leveraging an existing market, with existing players and existing liquidity. This allows Aera to by-pass almost all of the market mechanism criticisms about Futarchy (*e.g.* the cold start problem, lack of liquidity, lack of investor interest, etc).

Moreover, the submitters and arbitrageurs already exist today as DAO members, data science companies, and trading firms. Onboarding these users into this "3-sided" marketplace is much easier than starting a new futarchy market from scratch. Aera is using the existing market to solve a much more constrained problem with a much more clear objective function on a shorter time horizon, allowing for more rapid feedback loops and an effective culling of weaker players. Finally, Aera assumes that there exists a well-defined objective function, unlike traditional futarchy which requires a very extensive process of defining and refining the objective function.

Usage of CFMMs. CFMMs are by far the most popular form of decentralized trading mechanism used on blockchains. The most popular CFMM, Uniswap, has helped facilitate over \$1 trillion in trading volume since 2018 [25]. One of the main advantages of using CFMMs over other types of automated market makers is because computing optimal arbitrage is computationally simple. In particular, while the optimal arbitrage problem for CFMMs is convex and polynomial time [9], optimal arbitrage for traditional market scoring rules used in futarchy can be as complex as #P-hard [28]. This has been a vexing problem for both practical and theoretical applications of market scoring rules.

Another important reason for using CFMMs is that they naturally map to portfolios that are rebalanced by arbitrage. For instance, [39] demonstrated that geometric mean market makers are precisely portfolios of N-assets with weightings $\pi \in \{p \in \mathbf{R}_+^N : \mathbf{1}^T p = 1\}$ such that arbitrageurs ensure that the asset composition stays near π. Given that there are a number of large aggregators that route trades through CFMMs, such as 1inch and Matcha, arbitrage order flow to trade against a DAO's treasury portfolio is much easier to source with CFMMs. Finally, we note that CFMMs have stronger guarantees on price manipulability than market scoring rules. Manipulating prices in a CFMM will cause submitters to take larger losses than they should, so prevention of price manipulation is important for safe portfolio rebalancing. If one wants to manipulate the price on a Uniswap-style exchange by a factor of ϵ, *i.e.* move the price from p to $(1 \pm \epsilon)p$, then it will cost at least $\sqrt{\epsilon}L$, where L is the liquidity of the CFMM pool [140, App. E].

Time-scale separation. In futarchy, buyers and sellers within a prediction market can trade at any time until a particular market close time to allow for a government or DAO to make a decision [24, 61]. For example, consider a market for US GDP in 2032 where participants could trade a binary option on whether the US GDP is greater than or less than \$100T. Participants can trade freely between the positive and negative assets until 2027, at which point the DAO or government will execute a public policy based on how the market settles. The instrument can be thought of as something with a payoff of \$100 if the US GDP in 2032 is greater than \$100T or \$0 otherwise. The asset might be worth \$45 while the opposite share is \$55 and this price is allowed to freely fluctuate until 2027. On the other hand, Aera forces the submitter to calculate and theorize at time t about on-chain state at time $t + \delta$ and the submitter cannot change their stated outcomes or trade out as futarchy participants can.

Anyone can trade freely with the CFMM within Aera, but a submitter cannot collude with an arbitrageur to implicitly change their prediction without a large capital cost. To see why, suppose that a submitter tendered a portfolio with weights $(0.5, 0.5)$. However, at time $t + \delta/2$, the submitter has new information and realizes the optimal portfolio is $(0.4, 0.6)$. The submitter could perform trades herself (or collude with an arbitrageur) to make $f_{\text{DAO}}((0.5, 0.5), \Sigma_{t+\delta}) > f_{\text{DAO}}((0.4, 0.6), \Sigma_{t+\delta})$. This would, however, cost the submitter at least $\sqrt{\epsilon}L$ in a Uniswap-style CFMM and also cost extra in impermanent loss. Also note that this cost is linear in time, so that such a manipulation has a cost that is $\Omega(\delta)$ (See Appendix A for the generalization to generic CFMMs). Aera's collateral model and parameters (which is sketched in Appendix B) is constructed such that the cost of performing such a manipulation is greater than the fees earned by the staker which is sketched in Appendix A (and formally proved in a subsequent paper).

The time-scale separation between submitters and arbitrageurs is crucial to ensuring that this cost of manipulation is higher than the expected profit from fees. Futarchy has been critiqued for having a 'whale problem' where a single entity buys all of the tokens for an event happening right before it happens, reducing the profit of those with better information at earlier times [24, 65]. As the cost of manipulation is $\Omega(\delta)$ for Aera, the whale problem becomes less viable as the epoch length δ increases. We note that careful design of how and/when fees are paid out after an epoch has completed is required to ensure this cost is $\Omega(\delta)$ in practice.[8]

On-chain predictions only. As mentioned, submitters have to have non-trivial ability to calculate and theorize about on-chain state $\Sigma_{t+\delta}$ in order to be successful. However, it is quite important that the predictions are *only* about on-chain state. Prediction markets such as Augur [109] or Astraea [4] and collaborative trading markets like Numerai [35] generally try to solve the oracle problem, which involves using incentives to bring off-chain information on-chain. This, however, means that there is no purely objective way of evaluating if a given off-chain data point is valid outside of the consensus of market participants. Prediction markets restricted to on-chain bets only, such as Aera, can objectively evaluate if a particular prediction was correct or not simply by evaluating f_{DAO}. Moreover, grading how well a participant is doing relative to others for on-chain predictions is easier as the entire state necessary to compute a grade is available (provided data availability liveness of the layer 1 blockchain).

[8]As an example of a similar epoch-based mechanism that accidentally had $o(\delta)$ cost due to how fees were paid in the derivatives exchange dYdX, see [67]

Aera overcomes the notable failures of Futarchy. Most of the critiques against futarchy tend to deal with the whale problem, a lack of liquidity and/or ease of trading, and the difficulty in repeatedly grading users based on historical performance [24, 65]. By restricting the types of predictions that can be made and using CFMMs to have a high cost of manipulation, we argue that these features of Aera allow it to avoid the pitfalls of futarchy. While this paper will focus on the usage of Aera for treasury management (which maps cleanly onto CFMMs as a trade execution mechanism), one can imagine that the futarchy-like properties of Aera could help manage other on-chain state. DAOs use on-chain governance to manage risk parameters (*e.g.* margin requirements, interest rate curves) and incentive spend (*e.g.* incentives to get new users using a protocol) and it is likely that a modifcation of the treasury management mechanism could be used to have a market mechanism for updating these parameters.

Overview. The remainder of this paper is divided into seven sections. §2 describes the treasury management problem for DAOs that have protocol-provided insurance and §3 gives a breakdown of approaches to grading and aggregating parameter submissions necessary for such a mechanism to function in a decentralized manner. Subsequently, §4 provides replicating portfolio strategies that portfolio submitters can use within Aera. §5 and §6 describe the smart contract architecture and security audit for the protocol. §7 describes simulation and test vault performance results for these smart contracts while §8 describes future work and the protocol's long-term roadmap.

2 Improving Capital Efficiency for Protocols with Insurance Funds

To improve the capital efficiency of their protocols, risk bearing protocols such as Aave (lending) and Synthetix (derivatives) utilize stake-based insurance funds. Such funds aim to cover capital shortfalls due to cascading liquidations and, at least theoretically, allow the protocol to be more aggressive with risk parameters (such as margin requirements). A protocol's goal is to utilize the insurance fund to create more competitive margin requirements. This would allow such protocols to offer high amounts of leverage to users while providing roughly the same security against cascading liquidations.

Such treasury reserve funds involve holders of a protocol's native token who stake (or lock-up) this token in order to earn inflation rewards (which are denominated in the protocol's token). For instance, the Aave safety module allows Aave token holders to stake their Aave and earn nominal Aave-denominated yield. However, if the system ends up having positive liabilities (*e.g.* shortfall events in Aave), then the protocol can sell or distribute assets in the insurance fund to cover these liabilities. In these scenarios, users who staked assets into the insurance pool realize losses, in a manner similar to holders of traditional credit default swaps. These losses can also be thought of as claims payouts in traditional insurance, except that they are triggered by oracle updates to a protocol that cause insolvency (e.g. an event such as assets $<$ liabilities).

Currently, the majority of these insurance funds are treated as treasury assets which can be used to cover shortfall event payouts. However, these treasuries are mainly denominated in the protocol's native token with some small amount of reserve assets (*e.g.* fees accrued) that are collected in other assets such as ETH and stablecoins. If the liabilities within the protocol are not covered by the assets held within the insurance fund, then the protocol is in technical default. As such, we can view the problem of a protocol providing sufficient insurance to its users as a particular form of treasury management.

Insurance and Fund Management in DeFi. There have been a plethora of attempts at adding insurance to cryptocurrency protocols. One of the main drivers of insurance has been an attempt at creating an safe on-chain savings account, where depositors earn yield from borrowers who desire leverage. This would act much like government-backed deposit insurance (*i.e.* FDIC insurance) that protects savers when there are systemic collapses in the DeFi banking system. However, most attempts at DeFi protocols that aim to provide such a service focus on directly providing this to users. Bridge Mutual [96], Risk Harbor [117], InsurAce [137], Unslashed Finance [137], Cozy Finance [38], and Opyn V1 [49] are protocols that allow users to deposit a portion of their earned interest into an insurance pool that pays out on a liquidation event. Moreover, the largest protocol to act as a savings account, the Anchor protocol on the now-defunct Terra blockchain, failed spectacularly and many DeFi insurers did not pay out [98, 137].

The interface of forcing savers to buy their own deposit insurance is quite unnatural — in the off-chain world, the financial intermediary (e.g. a bank or exchange) purchases or holds insurance on customer fund losses. In traditional finance, this insurance aggregation effect allows for cheaper coverage to be provided to a larger number of users and participants. Can something similar be replicated in DeFi?

One way of replicating this is via rebalancing a DeFi protocol's stake-based insurance funds. If a portion of the assets in the stake-based insurance fund could be diverted to uncorrelated yield-bearing assets, then the protocol could better weather tail risks. This would effectively serve as an analogue of FDIC insurance, provided that the diverted assets were actively managed to avoid overconcentration and a loss of correlation to liabilities. We term any mechanism for rebalancing an insurance fund's assets to match its liabilities and/or hedge downside risk, *risk-aware treasury management*. We note that risk-aware treasury management differs from insurance in that the agents who provide the treasury management service are not utilizing their own capital to cover insolvencies but rather provide actuarial recommendations for rebalancing a DAO's insurance.

Mechanics of Protocol Insurance. In order to describe mechanisms for rebalancing an insurance fund's assets, we first need to understand how existing DeFi protocol insurance works. Suppose that a protocol such as Aave issues a native token T whose supply inflates by $i(h)$ units of T at block height h. Currently, insurance funds are structured such that holders of T can stake their assets in a smart contract to receive a pro-rata share of $i(h)$.

Specifically, if a user has locked $b\%$ of assets in the staked contract at block height h, they will receive $\frac{b}{100} \cdot i(h)$ units of $\textsc{t}$. On the other hand, if there is an insurance claim where the fund has a liability of $L(h)$ units of $\textsc{t}$, then the staker will lose $\frac{b}{100} \cdot L(h)$ units of $\textsc{t}$ at block height h.

During times of duress, like Black Thursday [136], protocols will have liabilities denominated in other currencies that can be larger than the amount held in the T-denominated insurance fund. This leads to the liability $L(h)$ growing dramatically around such events. Moreover, if price of asset $\textsc{t}$ also plummets at the same time, the liabilities will increase further. Finally, if liquidation cascades are caused due to sudden growth in $\textsc{t}$ denominated liabilities $L(h)$, then the T-denominated insurance fund can be drained.

Stake-based insurance funds work well when the cost of paying out the $n + 1$st claim (in units of $\textsc{t}$) doesn't depend strongly on the execution price of claim n. One way of covering for these scenarios is to swap a portion of the block inflation $i(h)$ into assets that are correlated with the liabilities $L(h)$. If it were possible for an on-chain investment vehicle to guarantee correlation to liabilities while also providing positive return, then DAOs can reduce their token inflation. Moreover, given the extra coverage for tail events provided by being correlated to $L(h)$, protocols can be more aggressive with regards to risk parameters, which improves capital efficiency.

2.1 Hedging Protocol Risks via Treasury Management

In this section, we describe Aera's strategies for rebalancing a DAO's treasury to ensure that there is sufficient correlation to liabilities. We first provide backgrounds on rebalancing vaults before describing the incentive and security properties of such a vault. The formal guarantees of this vault strategy will be described in a subsequent paper focused on game theoretic properties of Aera and in the appendices.

Risk-Aware Treasury Vaults. Aera builds on the success of on-chain reinvestment vehicles, such as Yearn Finance vaults, to achieve these goals. Consider a portfolio π of assets, including the protocol's native tokens $\textsc{t}$ and liquid assets that are strongly correlated to outstanding borrow (*e.g.* ETH, ETH options, and stablecoins). Aera will manage this portfolio in a *vault*, which handles the business logic for updating the portfolio and executing the rebalancing strategy.

Vaults are smart contracts that assist DAOs with the management of their portfolios of assets autonomously using a fixed strategy. There are a number of protocols that utilize vaults, which were initially pioneered by Yearn Finance. Within DeFi, the majority of vaults are utilized to execute so-called 'yield farming' strategies [10]. These strategies involve taking an initial portfolio π of assets, depositing the portfolio into various protocols that are offering token-denominated yields, and rebalancing the portfolio when yields change or deviate from a target return. Yield farming vaults have been used for options [56, 44], liquidity provision [43, 113], lending [106, 122], and managing perpetual derivative positions [112].

However, as the majority of vaults use transparent, public, and on-chain strategies, they are unable to adjust their strategy in response to losses due to adverse selection. Moreover, most strategies optimize simple objective functions, such as yield maximization, and are agnostic to the utility functions of the vault's depositors. Aera's risk-aware vault is not agnostic to its depositors' objectives, as the depositors are DAOs that have liabilities to cover. A depositing DAO's utility function is codified as an *objective function* which is used to grade submitters and determine their share of fees. The goal of the parameter submitters is to enable the strategy utilized on-chain to be dynamic and competitive, reducing the overall adverse selection and transaction cost for a DAO.

A unique challenge faced by vaults offering dynamic rebalance strategies is deciding on an optimal rebalancing frequency. If a vault rebalances too frequently, then it will realize losses from excessive transaction costs. Moreover, high-frequency rebalances will lead to excess selling of the protocol's native token T, reducing the price of T and limiting the overall coverage of liabilities in dollar or ETH terms. On the other hand, if the vault rebalances too slowly, then it will not be adjusting to asset correlations and the portfolio is unable to track the liabilities of the depositing protocol. Aera optimizes rebalancing between these two extremes by using a two-speed incentive mechanism that has a 'fast' and 'slow' path.

Time-Scale Separation. There are two time-scales within Aera: a 'slow' time scale that represents when a particular portfolio strategy is fixed and a 'fast' time-scale for arbitrageurs who trade against the vault to rebalance portfolios. The slow path in Aera involves parameter submitters who submit a portfolio π to the vault infrequently. Submission of parameters occurs on an epoch basis, where an epoch is defined to be a fixed number $E > 0$ of blocks. The kth epoch consists of blocks between heights kE and $(k+1)E$ and parameters for epoch $k+1$ are submitted prior to reaching block height $(k+1)E$. At epoch $e > 0$ and given a target portfolio π_e, the vault adjusts its portfolio from π_{e-1} to π_e via arbitrage.

This defines two time-scales: the epoch time E and the single block time (arbitrageurs can interact with the vault at any block height). One one hand, this separation of time-scales ensures that there is sufficient time for arbitrageurs to adjust the portfolio without causing excess price impact and/or payment of transaction fees (which would be realized if we traded from π_{e-1} to π_e in a single block, c.f. [118]). On the other hand, this time-scale controls how well we can grade parameter submitters for matching portfolio assets with liabilities. If we have too short of an epoch length E, then grading the submitters will be difficult as there will be too much noise in the rankings of submitters.[9] On the other hand, if the epoch length is too large, then the submitters will not be able to adjust the portfolio fast enough to changes in liabilities (which are exogenous to Aera).

In a decentralized, pseudonymous system, it is possible (and even likely) that the parameter submitters and arbitrageurs collude. Aera is designed such that even if the parameter submitters and arbitrageurs collude, any expected gains made by one party are at the expense of the other (e.g. the sum of their utilities is non-positive sum in expectation). However,

[9]As a simple demonstration of this, suppose the grades at each block of n submitters are drawn independently from distributions $D_1, \ldots, D_n$ with bounded moments. The net grade of a user can be defined as $g_i = \frac{1}{E}\sum_{h=1}^{E} d_{i,h}$, where $d_{i,h} \sim D_i$. This will converge to the true mean grade distribution at rate $O\left(\frac{\mathbf{E}[|d_{i,1}|^3]}{\mathbf{Var}[d_{i,1}]^{3/2}\sqrt{E}}\right)$ via the Berry-Esseen theorem. This illustrates that small epochs lead to noisier grades.

it is still possible for collusion to be profitable around particular rare events (*e.g.* persistent price shocks to the assets in the portfolio that last for many epochs). One mechanism for limiting the profitability during these events is to place limits on the maximum portfolio change within a single epoch.

Suppose parameter submitters tender portfolio weight changes that are too large relative to the optimal portfolio (*i.e.* the portfolio that optimizes the objective function). Then the vault will quote a trading price that is very off market and leak value to external arbitrageurs. This is particularly pronounced in when there is a single parameter submitter who may atomically trade in response to portfolio changes. To reduce the profitability of collusion, we introduce constraints that bound the maximum weight change for each asset, *i.e.* $|(\pi_{e+1})_i - (\pi_e)_i| < c_i$ for all assets i. Note that c_i can depend on properties of an asset's price trajectory, such as its volatility and jump size distribution. With the appropriate economic incentives associated with damaging submissions (see Appendix B), we are able to relax some of these constraints (e.g. increase c_i).

2.2 Participant Incentives

Parameter submitters are rewarded in AERA inflation for submitting risk parameters, whereas arbitrageurs sometimes receive AERA bonuses for trading against the vault. To receive rewards and earn fees, parameter submitters must put up capital on a particular vault in order to submit parameters to this vault.

Parameter Submitter Incentives. Parameter submitters in Aera are crucial to ensuring that DAO's have access to treasury risk management optimization. However, parameter submitters need to be disincentivized from providing bad parameters. In particular, submitter capital is put at risk of loss in order to used force submitters to 'put their money where their mouth is.' Parameter submitters are required to stake a sufficient quantity of capital to make submissions. If their submissions cause negative outcomes or losses for the vault, their staked assets are slashed proportional to the loss incurred.

As such, submitters are incentivized to enter the parameter submission contest only if they have sufficient confidence in their ability to predict future objective function values. However, a submitter's rewards will be higher when they have both higher confidence predictions and higher stake in the system. Thus submitters are incentivized to have better predictive models and data aggregation capabilities relative to others.

One other important metric for the protocol to monitor is the submission quality (over many epochs) of individual submitters The protocol aims to reward the highest quality submitters over time. A parameter submitter is graded based on their contribution to the improvement of the vault's objective function. Various methodologies for performing such grading will be described in §3. If a parameter submitter has a high 'quality score' (*e.g.* time average of their grades), their submissions will be weighted higher during the aggregation of portfolios tendered by submitters. This will increase their fee attribution measured via weighted Shapley values (see Appendix D) should they continue to have submissions that improve the vault's objective function. The game theoretic properties of using the weighted Shapley value for this iterated game will be analyzed in a follow-up paper.

We also note that oracle protocols such as Pyth [13] have learned quality scores for each submitter. Instead of ranking submitters based on how much they improve an objective function, the submitters in Pyth are rewarded based on how tightly they are able to provide confidence intervals on price feeds.

DAO Incentives. Depositing DAOs have the most clear incentives to analyze within the Aera protocol. These DAOs want the benefits of active asset management aligned to their utilities while still having credible decentralization. Given that DAO governance generally cannot handle active asset management, DAOs have a natural incentive to avoid losses in utility by using Aera. However, we also provide incentives to DAOs that deposit into vaults.

Arbitrageur Incentives. Aera's fast path utilizes arbitrage incentives to ensure that the portfolio stays within risk tolerances. This ensures that arbitrageurs will continue to interact with the vault even when the assets and/or the underlying blockchain are in times of duress. One can view any incentives paid to arbitrageurs as a form of liquidation incentive (akin to those in Aave, Compound, and Maker) used to ensure that the vault can reach its target portfolio within an epoch. These risk tolerances are utilized when grading staker performance — stakers are only slashed when arbitrageurs cause the DAO's portfolio to drift far enough from the target portfolio.

In Figure 3, we see an example of how risk parameters $r_h \in \mathbf{R}_+^2$ are utilized to control arbitrageur incentives. When the portfolio value (*e.g.* net asset value of the vault in numéraire terms) is within the interval $[r_h^-(e), r_h^+(e)]$, stakers receive all rewards for an epoch $i(e)$. However, if the portfolio value breaches either of these limits and arbitrageurs are able to extract $\alpha(e)$ units of profit (represented by the red shaded areas in Figure 3), this profit is taken from native token rewards and given to arbitrageurs. As such, stakers only realize $i(e) - \alpha(e)$ in total rewards for epoch e. If $i(e) < \alpha(e)$, then stakers are slashed, which effectively passes on the arbitrage loss to stakers. More details about arbitrageur and staker incentives can be found in Appendices B, C, and D.

3 Grading and Aggregation of Parameter Submissions

The core mechanism that Aera uses to disincentivize malicious parameter submission is via the allocation rewards proportional to the 'quality' of a parameter submitter's value. Newer oracles such as Pyth [13] also introduce a notion of rewarding users based on quality. However, measuring quality and attribution of submitters is difficult within a decentralized, pseudonymous environment. For instance, parameter submitters can collude with each other to maximize rewards while providing little to no value to the depositing DAO (see [52] for an example in Proof of Stake). Moreover, parameter submitters and arbitrageurs can collude to try to drain resources from the vault. Disincentivizing collusion by adjusting fees and/or slashings to each users is crucial to the success of Aera.

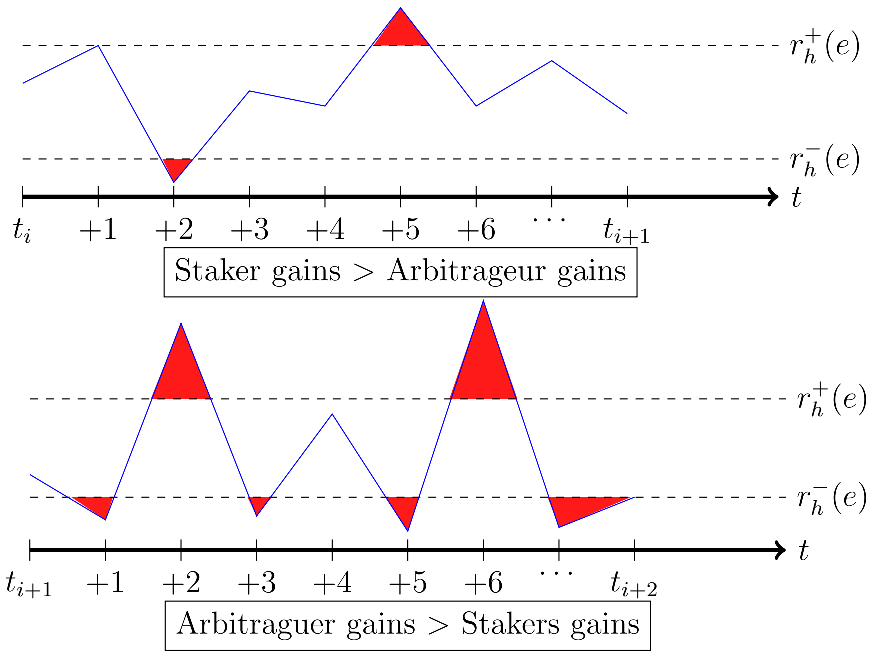

Figure 3: Evolution of a price trajectory (blue) and risk parameters $r_h(e) = (r_h^+(e), r_h^-(e))$. When the price leaves the rectangle bounded by $r_h^{\pm}$, arbitrageurs can trade against the vault. (Top) A price path and r_h with minimal losses to arbitrageurs (e.g. $i(e) > \alpha(e)$). (Bottom) A price path and r_h with many losses to arbitrageurs (e.g. $i(e) < \alpha(e)$). See Appendix C for more details.

We treat the role of a parameter submitter as similar to the 'expert' in expert learning problem of machine learning [74]. Expert learning refers to trying to learn the optimal value of an objective function f given a set of n predictions for the input that maximizes f. It can also be viewed as an online learning problem when there are repeated rounds of predictions, as there are in Aera. We draw from the expert learning and collusion resistance literatures to design the grading rules and rewards for parameter submitters. There are a number of considerations that need to be taken for any practical system such as computability, adversarial resistance, and speed of convergence to an optimum of f.

There are two main components of submission quality: grading and aggregation. Grading refers to the process of ranking the submitters based on the relative performance of their tendered portfolios. The goal of grading submitters is to ensure that the improvement to the DAO's objective function is attributed to them correctly. Model attribution has been long studied in machine learning, especially when studying the expert learning problem. Recent work in attribution for neural networks [86, 128, 127] and portfolio management at BlackRock [94] is the inspiration for our choice of grading mechanic.

On the other hand, aggregation is the process of taking a submitter's historical performance up until epoch e and using that to aggregate portfolios tendered at epoch $e + 1$. Aggregation is the main iterative step within Aera that forces submitters to care not only about their grade at a particular epoch e, but to be incentivized to have a history of grades at epochs $e' < e$. Parameter submitters build their reputation as a good DAO asset allocator by

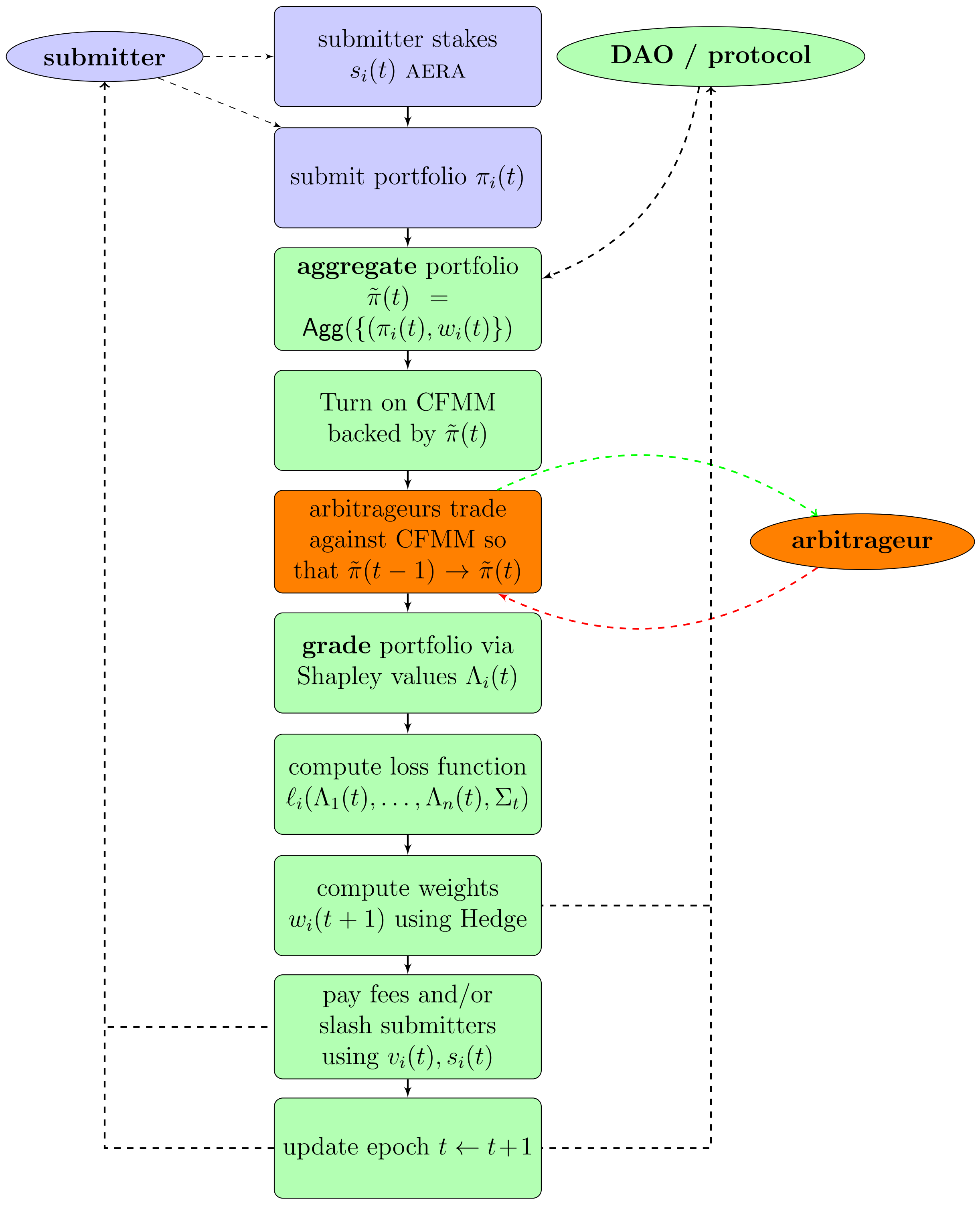
submitter
submitter stakes $s_i(t)$ AERA
DAO / protocol
submit portfolio $\pi_i(t)$
aggregate portfolio $\tilde{\pi}(t) = \mathsf{Agg}(\{(\pi_i(t), w_i(t)\})$
Turn on CFMM backed by $\tilde{\pi}(t)$
arbitrageurs trade against CFMM so that $\tilde{\pi}(t-1) \to \tilde{\pi}(t)$
arbitrageur
grade portfolio via Shapley values $\Lambda_i(t)$
compute loss function $\ell_i(\Lambda_1(t), \dots, \Lambda_n(t), \Sigma_t)$
compute weights $w_i(t+1)$ using Hedge
pay fees and/or slash submitters using $v_i(t), s_i(t)$
update epoch $t \leftarrow t+1$

Figure 4: Feedback loop for grading and aggregation of parameter submission values within an epoch. Rectangular boxes refer to smart contract calls within the Aera protocol, colored by one of the three agents that interacts with the protocol at that box. Dashed lines represent smart contract interactions by the DAO, submitter, and arbitrageur.

repeatedly earning good grades have their future portfolios weighted higher in the aggregate portfolio that the DAO trades into. By ensuring that submitters have the proper incentives to build good on-chain track records for various DAOs, Aera can develop a community of parameter submitters who can manage a DAO's treasury in a sustainable manner.

The entire flow diagram of the grading and aggregation process can be seen in Figure 4. One can see the different principle agents — the depositing DAO, the arbitrageur, and the parameter submitter — and the particular smart contract functions that they call. When reading this section, please refer to Figure 4 to understand where each component lies.

3.1 Grading Parameter Submissions

Parameter Submitter Collusion. One of the main problems with aggregating multiple user predictions when there is a reward is that users can collude to submit the same easy to compute but highly suboptimal answers. As an example, suppose that the objective function was a quadratic $f(x) = (x - a)^2$ where the value of a is unknown. parameter submitters could collude and all agree to submit the value $x = 0$, which leads to not learning any information about the value a. Adding in fees that parameter submitters compete for, however, can improve this situation by allowing users to find Schelling (or focal) points that lead to positive improvements towards reaching an objective. A major question in the design of Aera is the choice of fee mechanism that minimizes parameter submitter collusion.

In [52], it is shown that iterated Schelling point games in Proof of Stake that require an extra deposit (on top of existing stake) can be resilient to collusion. These games redistribute the deposits that stakers place after the outcome is decided. Unfortunately, the redistribution mechanisms used for the games in this paper assume that the deposit is the only currency involved and do not handle non-uniform redistribution. In the case of treasury parameter submission, one needs to reward users proportional to how much their submission increases the objective which is leads to non-uniform incentive distributions.

On the other hand, Shapley values are known to be able to quantify the amount of collusion amongst players even if they have a non-uniform distribution [60]. While Shapley values are expensive to compute, they provide the only credible mechanisms for measuring how much one needs to pay as an incentive to avoid collusion. Recall that the Shapley value can be thought of a measure of the expected impact of each users on the final output value of a game. If a user has higher expected impact, they can claim a higher percentage of fees (which incentivize non-collusion as users compete to increase their impact).

There are three important properties for fee attribution that one can derive from the Shapley value: fairness, correct baseline, and full attribution [94]. Fairness implies that if two submitters propose the same portfolio, then the fees attributed to them will be equivalent. Correct baseline value implies that the agents are graded relative to a fixed baseline. Full attribution means that all submitters have their values attributed (*e.g.* if I submit a non-trivial portfolio change, then there must be some fee attribution). These three properties ensure fair fee distribution to submitters based on their expected marginal contribution to the final portfolio.

The authors of [94], who use the Shapley value to attribute portfolio performance to particular features in a portfolio (which can be viewed as the submitters' individual tendered portfolios), demonstrate that simpler approximations miss the full attribution property. We also note similar properties are proved and analyzed in [126], which looks at the efficiacy of Shapley attribution for portfolio performance. Without the full attribution property, there is natural incentive to collude [94, §3.1]. Given that approximations of the Shapley value perform well at historical portfolio attribution, we utilize this as our main means for reducing parameter submitter collusion [94] (see Appendix D for more details on implementation).

Parameter Submitter and Arbitrageur Collusion. On the other hand, parameter submitters could collude with arbitrageurs (or be arbitrageurs themselves) and choose parameters that actively worsen the objective function value. When the portfolio realizes a loss and it can be attributed to a subset of parameter submitters, slashing that user's deposit by an amount proportional to the loss caused serves as a disincentive for submitters to collude with arbitraguers. Provided that the assets staked are sufficiently decorrelated from the assets in the vault, slashing is effective at ensuring that colluding parties have negative expected value.

However, there are scenarios where the staked assets are overly correlated to the vault assets where it is possible for the expected value of a colluding submitter and arbitrageur to be positive. We provide a sufficient condition in our follow-on paper that provides sufficient conditions for ensuring that the probability that this event happens is small. This sufficient condition is used to adaptively adjust the block reward and fee distribution such that the likelihood of a parameter submitter profitably colluding with an arbitrageurs is small and can be controlled via protocol governance.

Finally, we note that we separate the value of staked assets deposited by a parameter submitter from the agent's quality score. The staked value determines the pro-rata share of AERA rewards captured by the parameter submitter whereas the quality score determines the fees that are distributed to that particular submitter.[10] The protocol does not allow quality scores to be transferred between different Aera vaults in order to avoid issues where user's with a high quality score in one vault utilize it to collude with arbitrageurs in another vault. This also allows DAOs to help seed the community of parameter submitters to an Aera vault that they deposit to with members of their risk community.

3.2 Aggregating Parameter Submissions

While one aspect of grading involves computing how fees should be distributed to minimize collusion, another aspect involves aggregating the portfolios tendered by parameter submitters to construct the DAO's portfolio for the next epoch. Formally, this means that we need to construct a series of weights $w_1(h), \dots, w_n(h)$ for each of the n parameter submitters

[10] We note that the separation of fee and value distributions also exists within auction design. The only computationally credible, truthful, and bounded communication auction [42] utilizes deposits and slashing where deposits act as fees to ensure that the auctioneer cannot create fake bids to increase the final sale price. Similarly, we utilize the fee redistribution to ensure that large stakeholders cannot influence the final aggregated portfolio unless they also have a high quality score (measured via the Shapley value)

at block height h. The weights represent the confidence that the protocol has in the ith submitter based on their prior tendered portfolios. Constructing a baseline portfolio can be thought of as a multi-armed bandit problem (which will be formalized below): the k parameter submitters are arms in the bandit problem and the protocol aims to learn a set of weights $w_1, \dots, w_n$ based on prior performance. Finally, note that a choice of portfolio aggregation mechanism also impacts fee distribution, as utilizing the Shapley value relies on selection of a baseline portfolio (see [94, §2] for further discussion on this).

There exist a number of algorithms for solving multi-armed bandit problems under both stochastic and adversarial assumptions. All algorithms known to the authors have different computational, accuracy, and convergence trade-offs. First, we will make the connection between the portfolio aggregation problem and bandit learning algorithms. Then we will review three main mechanisms for solving bandit problems: Hedge algorithm, Follow-the-Leader, and convex aggregation.

Multi-armed Bandits. The multi-armed bandit problem involves a learner who aims to maximize their expected value from sampling a loss function ℓ given a time budget T. This loss function can only be sampled by the learner pulling one of n arms to receive a loss ℓ_t at time t. A learner creates a distribution $w_t \in [0,1]^n, t \in [T]$, samples an arm $i \sim w_t$ and then pulls arm i to realize loss ℓ_{t+1}. The algorithmic goal of a multi-armed bandit algorithm is to learn a series of distributions $w_t \in [0,1]^n, t \in [T]$ utilizing the observed losses $\ell_1, \dots, \ell_t$ such that $\mathbf{E}_{w_t}[\sum_t \ell_t]$ is maximized.

We can view the portfolio aggregation problem as having a similar structure to a bandit problem[11] — the learner is the protocol and the parameter submitters are the arms. Our losses can be viewed as portfolio performance for each parameter submitter, each of which submits a portfolio $\pi_i(t)$. Portfolios are graded using an objective function ℓ, which can represent yield, diversification, or risk. In our initial usage, the function ℓ will be a function of the Shapley value of the portfolio submission. At each discrete epoch t, the protocol's portfolio is equal to $\Pi(t) = \sum_{i=1}^n w_i(t)\pi_i(t)$. Define the optimal weights in hindsight to be $w^*(t) = \arg\max_w \ell\left(\sum_{i=1}^n w_i(t)\pi_i(t)\right)$ and the optimal portfolio as $\Pi(t) = \sum_{i=1}^n w^*(t)\pi_i(t)$. If we define the loss realized for choosing weights $w_i(t)$ as $\ell_t = \ell(\Pi^*(t)) - \ell(\Pi(t))$, then we see that the allocation problem maps to a multi-armed bandit problem.[12]

There exist a number of algorithms for solving the multi-armed bandit problem under different assumptions. These assumptions include the i.i.d. scenario, when the UCB algorithm suffices [123], the generic stochastic setting [37], and the adversarial setting [14]. The two main algorithms for solving bandit-like problems in the stochastic and adversarial settings are the Follow-the-Leader and Hedge algorithms, respectively.

[11]Note, however, that the on-chain state and rebalancing mechanism can be introspected by vault guardians we don't expect them to treat the problem as a black box bandit problem

[12]The precise equivalence between the forcecasting problem where users provide probability distributions (like a portfolio) versus point estimates (like the standard bandit) is provided in [7] and [26]

Hedge Algorithms. The Hedge algorithm (also known as the multiplicative weights algorithm) constructs weights using an iterations of the following form

$$w_i(t) = w_i(t-1)\frac{e^{-\epsilon(t)\ell(\pi_i(t))}}{\sum_{j=1}^k e^{-\epsilon(t)\ell(\pi_i(t))}}$$

where $\epsilon(t)$ is a step size parameter. The multiplicative weight algorithm is always able to provide regret bounds of the form $\mathbf{E}_{w_i(t)}[\sum_t \ell_t] - \max_w \mathbf{E}_w[\sum_t \ell_t] = \tilde{O}(\sqrt{t})$, even in adversarial settings provided that ℓ is convex and/or is Lipschitz [14]. In some cases, it can be combined with other algorithms to provide $O(\log t)$ regret bounds, but only in restrictive stochastic (and non-adversarial settings).

One of the main difficulties with using the Hedge algorithm is tuning the step size parameter $\epsilon(t)$. The authors of [26] provided the first mechanism for adaptively selecting $\epsilon(t)$ in the following manner:

$$M_t = \max_{s \leq t} \max_{i \in [k]} 2^{\lceil \log_2 |\ell(\pi_i(s))| \rceil} \qquad\qquad \epsilon(t) = \frac{1}{2M_t}$$

Note that $\epsilon(t)$ is weakly monotonically decreasing as M_t is (weakly) monotonically increasing[13] and that $\epsilon(t) > \epsilon(t-1)$ precisely when we arrive at a new maximum (e.g. t is such that $|\ell(\pi_i(t))| > |\ell(\pi_i(s))|$ for all $s < t$). This is able to produce sharp regret bounds that interpolate between $O(\sqrt{T \ln k})$ in the case where $\max_i \ell(\pi_i(t))$ differs significantly from $\frac{1}{n}\sum_{i=1}^k \ell(\pi_i(t))$ and $o(\sqrt{T})$ if $\max_i \ell(\pi_i(t)) = \Theta\left(\frac{1}{n}\sum_{i=1}^k \ell(\pi_i(t))\right)$. We study the precise regret bounds of Aera further in Appendix B.

For the portfolio aggregation problem, this states that if the performances of the submitters' portfolios are similar, then we can stay close to the optimal portfolio. On the other hand, if the performances of submitters' portfolios varies wildly, then it takes us $O(\sqrt{T \ln k})$ epochs to readjust. The latter fact represents the adversarial resistance of the Hedge algorithm (*e.g.* an adversary cannot force linear regret on the protocol), but is pessimistic for the average case.

Follow-The-Leader Algorithms. Follow-the-Leader algorithms effectively do what they say: they place all of the weight $w_i(t)$ on the submitter with the best performance. Formally, this means that the weights are defined as

$$w_i(t) = \begin{cases} \frac{1}{|\arg\max_i \ell(\pi_i(t))|} & i \in \arg\max_i \ell(\pi_i(t)) \\ 0 & \text{otherwise} \end{cases}$$

In other words, this rule uniformly places all of the weight on parameter submitters who achieved the highest portfolio grade in the last round. While this algorithm performs well for stochastic inputs (*e.g.* the submitters draw $\pi_i(t)$ from a static, stationary, but unknown distribution) but is not adversarially resistant. AdaHedge [37] constructs an algorithm that

[13]Rules of this form are known as the 'doubling trick' [26, 37]

oscillates between Follow-the-Leader algorithms and the Hedge algorithm and uses a different weight update rule $\epsilon(t)$. The advantages of combining the Hedge algorithm with Follow-the-Leader algorithms is that one can improve portfolio performance when there is more consensus amongst parameter submitters.

Convex Aggregation and Federated Learning. Recent work on forecast aggregation has noted that for particular loss functions, there exists an optimal aggregation rule for particular loss functions if one pays submitters [99, 101]. This work, in particular, only works for convex functions whose duals can be efficiently computed. These methods provide precise payments needed to ensure that aggregation and collusion resistance are optimal without needing Shapley values. While this is currently intractable for the grading functions that Aera aims to utilize, simple functions like logarithmic loss (e.g. $\ell(\pi_i(t)) = \max_i \log(\pi_i(t)/\pi_0(t))$, where $\pi_0(t)$ is a benchmark portfolio) can be handled via these methods. We mention these methods as Aera could utilize them in a future version.

Another form of convex aggregation comes from federated learning. Robust aggregation in federated learning [111] solves a similar problem to portfolio aggregation. Federated learning involves aggregating the model parameters trained from multiple local models to generate a population-wide parameter set. As an example, linguistic models for auto-correct on Android phones are trained locally on users' phones, which are then aggregated to generate a baseline model that is installed on new phones [62]. Since these systems have been deployed to millions of devices, adversarial resistance is of critical importance when constructing aggregation algorithms.

The main difference between robust aggregation in federated learning and traditional bandit algorithms is that the aggregation goes from having the portfolio defined as a weighted mean (*i.e.* $\Pi(t) = \mathbf{E}_w[\pi_i(t)]$) to utilizing weighted medians (*i.e.* $\Pi(t) = \mathsf{Median}_w[\pi_i(t)]$). Formally, there have been a number of results that proved better adversarial stability for parameter aggregation with geometric medians [111] and repeated medianization [47]. Both of these aggregation methods involve constructing convex surrogates of a weighted median.[14] On the other hand, weighted medians and trimmed means (*e.g.* means where some fraction of outliers are ignored) have had the best adversarial resistance in live systems that had hundreds to thousands of parameter submitters [40, 82]. In particular, weighted median aggregation models outperformed trimmed means when undergoing targeted aggregation attacks for machine learning model parameters.

Rank-based Aggregation methods One other issue with all of the aggregation methods described above is that they focus on absolute ranking of submissions. For instance, the Hedge algorithm computes weights such that the ith agents weights only depend on their

[14]Recall that the w-weighted median of points $\alpha_1, \ldots, \alpha_n$ is defined to be a minimizer of $g(v) = \sum_{i=1}^n w_i \|v - \alpha_i\|$. If the weights are all equal to $\frac{1}{2}$, we recover the standard definition of a median. Note that this function is not generically convex (and hence doesn't have a unique minima), but is close to convex (*e.g.* for most continuous probability distributions over w, α, $\mathbf{Prob}[\|g - g^c\| < \epsilon] \geq 1 - \delta$ where g^c is the convex hull of g) and can be minimized via local methods, such as gradient descent [111]. We note that Pyth's median aggregation rule is a subset of the models described in [47].

submitted parameter. On the other hand, relative rankings such as ELO [135], Glicko [54], and α-rank [95, 139] compare the ith agent's performance with all of the other agents. There have been modifications of the Hedge algorithm that incorporate rankings, such as pseudo-ELO [68]. These modifications allow for an aggregation rule to tune between absolute and relative weightings.

Aggregation within Aera. Given the large design space for portfolio aggregation within Aera, our main focus will be placed on using simple to implement and audit aggregation rules. We will initially construct portfolio weights $w_i(t)$ initially using the Hedge algorithm with dynamic weights, as this is relatively simple to audit and backtest. Later versions of the protocol will try to utilize hybrid methods for weight construction, such as AdaHedge, and will use modifications to include relative rankings. Our weights will then be used to construct a portfolio aggregation using a robust weighted median method, similar to those used in Federated Learning and oracles like Chainlink and Pyth. Finally, our loss function will utilize a combination of individual users' portfolio performances and moving averages of their Shapley values.

4 Hedging Protocol Risk in DeFi

In sections 2 and 3 we looked at how an Aera vault can be rebalanced to reach a set of target portfolio weights. We also explored how a set of parameter submitters can be incentivized to submit portfolio weights that optimize a pre-defined objective function f. In this section, we connect the general Aera mechanism to the protocol risk use case by constructing an appropriate objective function.

We also look at how parameter submitters may construct competitive submission strategies. In particular, we provide a high-level description of how submitters can construct portfolios that are correlated to a protocol's liabilities. These models provide intuition for how submitters can construct their strategies and backtest them off-chain. In practice, submitters will likely modify these simple replication strategies to take advantage of other information (*e.g.* liquidity elsewhere in DeFi and off-chain). However, the idealized no-arbitrage portfolios, which are constructed by hedging a loan book's liabilities with options, provide a starting point for submitters to build their strategies.

Considerations in objective function selection. Parameter submitter incentives are derived from the objective function which necessitates care in its selection. One consideration is ease of calculation. The objective function or its derivative will need to be computed both for the purposes of parameter submitter assessment and likely in the work of parameter submitters. However, parameter submitters can at least circumvent complex objective functions by working with derived versions of the objective function that may be simpler to calculate. This computation should also be adaptable to the intended universe of Aera customers. For hedging lending protocol risk, data not easily indexed for different lending protocols may be avoided in favor of more readily and uniformly available data. Since we expect a correlation between the costs of parameter submission and the variety and quality of parameter submitters, allowing parameter submitters to reuse existing infrastructure for data gathering across multiple clients is an important long-term objective of this effort. Another concern is time based trade-offs. In our context, an objective function that is purely focused on yield may not capture the costs of paying for shortfalls denominated in specific liabilities. However, an objective function that is purely focused on matching liabilities may do so at the cost of long-term portfolio growth. Thus, the objective function should aim to capture the inherent balance between short-term needs and long-term portfolio sustainability.

Liability matching. For hedging protocol risk, we aim to assess future protocol liabilities by denomination and match the portfolio composition to the liabilities. Our goals in doing so are twofold. For one, readily having assets available to meet shortfalls in the correct denomination will minimize the need for forced trading and slippage costs. More importantly, however, it allows the portfolio to move in lockstep with the liability exposure during volatile markets without the need to predict individual asset price performance. In order to break down lending protocol liabilities by asset denomination, we turn to replicating portfolios.

Replicating Portfolios. Recent work [32] has demonstrated that under no-arbitrage, it is possible to replicate the exposure of each individual loan held by a DeFi lender such as Compound or Aave using barrier options. In this section, we will generalize these results to a portfolio of liens, providing theoretical guarantees for hedges of entire protocols. Note that the results of [32] generalize to lenders of constant function market makers (CFMMs) [9]. Since any payoff in DeFi can be replicated via lending of CFMMs [11, 32], extending these results to portfolios of loans provides a way to hedge any DeFi protocol's underlying risk. Using these explicit hedges, we will further describe a method for computing weights $w_i(t)$ from empirical data. While our results are optimistic as they rely on no-arbitrage, these results provide a basis for more accurate weight calculations that can be estimated using using Monte Carlo methods and Agent-Based Simulations.

The exposure held by a single loan can be replicated by a portfolio consisting of the collateral asset and two short barrier options. The first barrier option is a digital option that is sold to the borrower. Exercise of this option corresponds to the borrower closing their lien, as the protocol returns the collateral to the borrower in exchange for their interest payment and borrowed quantity. On the other hand, the second barrier option is sold to the liquidator and is only active when the value of the collateral asset is lower than the borrowed asset (in numéraire terms). The price level (which acts as a barrier) when this occurs is termed the *liquidation price*. To incentivize liquidators to exercise this option, protocols set the strike price of the option to be at a discount to the liquidation price. Certain protocols use a fixed strike price discount (Aave, Compound) whereas others have a dynamic, floating discount (MakerDAO, Liquity).

4.1 Lending Portfolio Risk

Suppose that we have a single loan where q_b units of currency are borrowed against q_c units of collateral currency. Let p_0 be the price of collateral currency in units of the borrowed currency. For instance, if the loan involved borrowing USDC against ETH collateral, p_0 would be the price of ETH in USDC terms. There are two further parameters that are used in on-chain lending: a reduced loan-to-value ratio ℓ_{bc} and a liquidation incentive a_{bc}. Note that these parameters might be a function of time (e.g. adjusted by governance or a risk manager), volatility, and/or be constant. For simplicity, in the the remainder of the exposition, we will assume they are constant. In practice, we deal with dynamic parameter choices by using simulation and numerical techniques as opposed to analytical models.

Given a collateral price p, we can write the protocol's exposure for a loan L_{bc} as [32, §4]:

$$L_{bc}(q_b, q_c, p_0, p) = q_c\left(1 - DO(\ell_{bc}p_0, p) - DOIC((1-a_{bc})\ell_{bc}p_0, p, \ell p_0)\right)$$

where $DO(K, H)$ is the payoff of a digital option at strike K and barrier H and $DOIC(K, S, H)$ is the payoff of a down-and-in (knock-in) barrier option at strike K, price S, barrier H. The risky exposure of the protocol is equivalent to the options exposure, e.g. $R_{bc}(t) = L_{bc}(q_b, q_c, p_0, p(t)) - q_c + \gamma(p(t))$, where $\gamma(p(t))$ is the interest accrued by the protocol. Let $p(t)$ be the price of collateral (in borrowed asset terms) at time t and define $\mathcal{L}_{bc}(t) = \{(q_b, q_c, p_0, p(t))\}$ to be the set of outstanding loans in the protocol. The protocol's total risk exposure E_{bc} to being long collateral c and short borrowed asset b is then

$$E_{bc}(t) = \sum_{(q_b,q_c,p_0,p(t))\in\mathcal{L}_{bc}(t)} R_{bc}(q_b, q_c, p_0, p(t))$$

If the set of assets held by the protocol is denoted $\mathcal{A}$, then the protocol's net risk exposure at time t, $\mathsf{E}(t)$, is

$$\mathsf{E}(t) = \sum_{b\in\mathcal{A}}\sum_{c\in\mathcal{A}} E_{bc}(t)$$

To hedge this risk, we need to consider a 'small' set of assets, $\mathcal{N}$ that we can use to neutralize the risk of $\mathsf{E}(t)$. Formally, given $\mathcal{N} \subset \mathcal{A}$ with $\mathcal{N} = o(1)$ in the number of assets $n = |\mathcal{A}|$, our goal is to find a series of portfolios $\pi(t) \in \mathbf{R}_+^{\mathcal{N}}$ such that

$$p_n(t)^T\pi(t) + \mathsf{E}(t) \approx 0 \tag{1}$$

where $p_n(t) \in \mathbf{R}_+^{\mathcal{N}}$ is the price of hedging assets (e.g. USD, ETH, ETH options). Equation (1) says that the hedging portfolio π approximately hedges the risk exposure of the protocol.

Before describing methods for estimating π, let's first inspect equation (1). When $p_n(t)^T\pi(t) > \mathsf{E}(t)$, the hedging portfolio is too big, meaning that while it can hedge the risks $\mathsf{E}(t)$, it is capital inefficient. On the other hand, when $p_n(t)^T\pi(t) < \mathsf{E}(t)$, the hedging portfolio cannot cover the risk exposure $\mathsf{E}(t)$ and more assets and/or a rebalance is needed. The latter scenario is far more dangerous than the latter scenario, so any mechanism to estimate $\pi(t)$ needs to penalize portfolios $\pi(t)$ such that $p_n(t)^T\pi(t) < \mathsf{E}(t)$ is less likely. In the future, we will likely modify this formula have some forecast on p_n at some future $t' > t$ to allow for making the requisite adjustments for factors such as oracle price validity and liquidity, but for the time being, we will assume that an agent has a simple protocol wealth function is given by $w(\pi, t) = p_n(t)^T \pi(t)$.

Objective Functions for Estimating $\pi(t)$. In this section, we describe the high-level intuition for selecting objective functions for portfolio construction. A more granular formulation for basic portfolio construction (*e.g.* selecting $\pi(t)$) and an objective function to evaluate performance of $\pi(t)$) is in Appendix E. In general, it is infeasible to fully decompose risk and returns without constructing an explicit utility function.[15] A common function in economic literature is isoelastic utility (a hyperbolic absolute risk aversion function) defined as

$$U(w) = \begin{cases} \frac{w^{1-\eta}-1}{1-\eta} & \eta \geq 0, \eta \neq 1 \\ \ln(w) & \eta = 1 \end{cases} \tag{2}$$

where w represents protocol wealth and η is a nonnegative constant increasing with risk aversion. Another is exponential utility defined as

$$U(w) = \begin{cases} \frac{1-e^{-\alpha w}}{\alpha} & \alpha \neq 0 \\ w & \alpha = 0 \end{cases} \tag{3}$$

where α is a constant increasing with risk aversion. The remainder of the discussion will be agnostic to the choice of utility function used.

Another important function to define is the rebalance cost function denoted as C where $C(\pi, \pi', \Delta_{t_r})$ denotes the wealth cost of rebalancing from existing portfolio π to a new portfolio π' efficiently over the time interval Δ_{t_r}. This cost function is commonly used when estimating portfolio performance and transaction costs in statistical arbitrage portfolios and ETFs [116, 36, 15]. One such function C could be given by

$$C(\pi, \pi', \Delta_{t_r}) = \sum_i I_i \sigma_i \left(\frac{|\pi' - \pi|_i}{\Delta_{t_r} \mathrm{ADV}_i} \right)^{\beta_i} \Delta_{t_r} |\pi' - \pi|_i p_i \tag{4}$$

where I is a scalar measuring arrival time intensity, σ is the annualized volatility, β is a concavity parameter, ADV denotes the average daily traded volume, p denotes price, the units of Δ_{t_r} are in days and i signifies the ith asset in π. Various forms of such models have been studied by Bouchaud [63] and Avellaneda [16, 134].

It then follows a myopic portfolio π' can be solved for using the following optimization problem:

$$\pi^* = \underset{\pi' \in \mathbf{R}_+^{\mathcal{N}}}{\arg\max} \; \mathbf{E} \left[U\big(w(\pi', t + \Delta_{t_r}) - C(\pi, \pi', \Delta_{t_r}) \big) \right] \tag{5}$$

[15]Being model-free means that one is able to measure the cumulative impact of risk aversion on wealth without an explicit utility function for wealth. Implicitly, this would involve modeling the arrival process of decision opportunities and computing the expected value gap between the Kelly criterion and a heuristic based on U. This would provide a direct empirical separation between risk and reward in these systems; however, the convergence rate of such a measurement would be too slow for blockchain environments.

which simplifies to

$$\pi^* = \arg\max_{\pi' \in \mathbf{R}_+^{\mathcal{N}}} \mathbf{E}\left[U\big(p_n(t)^T \pi'(t) - C(\pi, \pi', \Delta_{t_r})\big)\right]$$

if one makes the assumption that price returns behave as martingales.

To better account for the optimization depth we can introduce another time parameter Δ_t which serves as an evaluation window to optimize the expression over. With the assumption that $\Delta_{t_r} \ll \Delta_t$ so C is effectively a constant with respect to T within the integrand we can then define

$$\pi^* = \arg\max_{\pi' \in \mathbf{R}_+^{\mathcal{N}}} \int_t^{t+\Delta_t} \mathbf{E}\left[U\big(w(\pi', T + \Delta_{t_r}) - C(\pi, \pi', \Delta_{t_r})\big)\right] dT \tag{6}$$

where tuning Δ_t depends on the signal to noise tradeoff of our forecasts for w and C.

In practice this is a difficult expectation to compute since the outcomes are path dependent on the price process p. If the price process is (roughly) stationary, however, one can construct an estimate from π^* via Monte Carlo sampling of a stochastic differential equation (SDE). For completeness, we will describe this for a geometric Brownian motion, but note that other adapted processes (including those with jumps) can be utilized in this framework.

Suppose that we have estimates for S, ρ, μ, σ where S denotes current prices, ρ denotes the correlation matrix of the price returns, μ denotes the annualized drift and σ denotes the annualized volatility of the price process then we can generate samples [16] of a multivariate correlated geometric Brownian motion where each price path obeys the SDE given by

$$dS_t^i = \mu_i S_t^i dt + \sigma_t^i S_t^i dW_t^i$$

where the Wiener processes W are correlated so

$$\mathbf{E}\left[dW_t^i dW_t^j\right] = \rho_{i,j}$$

and approximate by discretely averaging over the samples. There are further dependencies on the price trajectory as well since the path will influence the decisions of the various agents e.g. liquidators, borrowers, suppliers, cascading profiteers which will lead to changes in the overall loan book $\mathcal{L}$ and market liquidity. If Δ_t is large enough updates to the protocol risk parameters l (loan-to-value) and a (liquidation incentive) will impact the value of π^* as well. Combining a fit price trajectory S_t, models for the different agent behaviors, and Monte Carlo simulation of the impact of the price trajectory on a user, one can numerically estimate π^* and submit it to the network. We note that there are a number of numerical idiosyncrasies with Ethereum smart contracts that make the simulation of such portfolios a delicate computation; see §7 for more details.

[16] One may need to do some likelihood normalization given the empirical variances and sample sizes

Comparison to Portfolio Insurance. This above formulation has parallels to Merton's portfolio problem [91, 92] within intertemporal portfolio choice where at every time t there is a choice for consumption c_t along with a fraction of wealth to invest in the market π_t with the remainder in a risk-free asset. The objective is given by

$$\sup_{w_t, c_t} \mathbf{E}\left[\epsilon^{\eta} e^{-qT} U(w_T) + (1-\epsilon^{\eta}) \int_0^T e^{-qt} U(w_t - c_t)\, dt\right] \tag{7}$$

where q denotes a discount rate, w_T denotes terminal wealth at time T, c_t denotes consumption and ϵ parameterizes level of bequest and U denotes the constant relative risk aversion utility function given by

$$U(w) = \frac{w^{1-\eta}}{1-\eta}$$

and η is the risk aversion constant and the wealth w adheres to the stochastic differential equation

$$dw_t = [(r + \pi_t(\mu - r))w_t - c_t]\, dt + w_t \pi_t \sigma\, dW_t$$

where r denotes the risk-free rate, μ is the annualized drift and σ is the annualized volatility of the market and W_t denotes the Wiener process. We note that one also sees this formulation used in practice with Wealthfront's investing guide [131] which demonstrates different portfolio allocation processes according to risk appetite.

Equation (7) is often solved when trying to estimate how much portfolio insurance one should acquire [79]. Portfolio insurance refers to any additional products, such as credit-default swaps or options, that are added to a portfolio to either increase the solutions to equation 7 or reduces its variance. Such insurance, at least in its original form, is used sparingly in modern finance as liquidity for derivative and spot assets has fragmented causing risk models to have inherent uncertainty [80, 103]. However, in the case of DAO treasury management, the DAO often is the primary source of liquidity for both spot and derivative liquidity for its native token. As such, solving portfolio insurance problems like equation (7) can be performed more easily in DeFi, especially if a DAO's portfolio can include being a dominant liquidity provider within the derivatives market. We note that V2 of Aera, which will support generic ERC-4626 assets and far out-of-the-money put options (also termed crash puts) will allow a DAO to diversify its portfolio by dynamically creating liquidity for derivative products based on its treasury assets (see §8).

5 Vault Structure

5.1 Proposed design of Aera vault

Our initial proposed design of an N-asset Aera vault rebalances treasury portfolios using a CFMM and a single trusted parameter submitter, but does not aim to tackle aggregation and/or slashing incentives. The underlying CFMM vehicle is a controlled (unfinalized) Balancer pool. Additional features are developed to support deposits and withdrawal of capital for the shortfall reinsurance use case. It is expected that future iterations of the vault will add the capability for multiple parameter submitters and for more robust incentives/slashing as well as support for delegation in this order.

We have developed an implementation of a fully functional treasury risk management reserve vault using the above constraints (V1) and another version is in development (V2). V1 is comprehensively audited and running in production on the Polygon network.

Aspect	Implementation
Asset universe, U	ERC20 tokens only; can be restricted to $U = \{\text{ETH}, \text{USDC}\}$
Submitter set	Single submitter, $n = 1$
Incentives	Linear management fee accrued to submitter
Parameter aggregation, Agg	Single submitter, $w_i = \mathbf{1}_{i=0}$
Epoch duration, E	Determined for each submission
Slow path	Submit weights for linear rebalancing schedule using Balancer
Fast path	Vault is traded on Balancer as weights move towards target
Safety features	Trade pausing
Risk specific	Weight-invariant withdrawals and deposits
	Withdrawal validator module
	Notice period to withdraw all funds

Table 1: V1, Single submitter N-token vault

Aspect	Implementation
Asset universe, U	ERC20 tokens and ERC4626 vault shares
Submitter set	Single submitter, $n = 1$
Incentives	Linear management fee accrued to submitter
Parameter aggregation, Agg	Single submitter, $w_i = \mathbf{1}_{i=0}$
Epoch duration, E	Determined for each submission
Slow path	Weight submission for ERC20 tokens, minting for ERC4626 shares
Fast path	Vault is traded on Balancer as weights move towards target
Safety features	Trade pausing and oracle-protected resuming
Risk specific	Oracle-protected withdrawals and deposits
	Withdrawal validator module
	Remove notice period, introduce minimum earned fee

Table 2: V2, Single submitter N-token or yield-bearing token vault

In each vault, we distinguish two privileged roles that have special access to the Area vault:

- The **vault owner** (client protocol), which provides, withdraws funds and can give notice for vault termination
- The **vault guardian**, a single parameter submitter that receives an optional fee in return

Each vault also has a **withdrawal policy** ($\mathcal{W}$), a function from the set of possible on-chain events to combinations of token amounts that can be withdrawn $\mathcal{E} \rightarrow \mathbb{Z}_{256}^{N}$. One or more components of $\mathcal{W}$ will be positive in the event of a re-insured shortfall event.

Withdrawals are implemented as a Balancer `updatePoolBalance` operation with the Balancer spot price kept constant. The pool also supports enabling and disabling trading and withdrawals/deposits are only recommended in a disabled state to mitigate MEV exposure. Balancer weights are determined via gradual adjustment using `updateWeightsGradually`. Weights change continuously and are recalculated on every trade.

The current Balancer weights will deviate from the target weight value in two circumstances:

- Following target weight submission
- Following asset withdrawals.

The single submitter controls two other parameters in addition to the vault weights: the **swap fee** and the duration for each weight update. We would expect the submitter to increase swap fees in periods of higher anticipated impermanent loss and decrease it in periods of relative asset price stability. We expect the submitter to calibrate convergence speed according to the overall size of the treasury and trading volume.

5.2 Slow and fast path rebalancing

The Aera vault needs to be able to cheaply re-balance the reserve holdings in response to both withdrawals and parameter submissions. The proposed Aera design is inspired by two existing mechanisms for treasury re-balancing, which we detail briefly.

5.2.1 Balancer pool with controllable weights

The first model is a pure Balancer pool with weights reflecting the preferred allocation of assets, a popular choice for existing mixed-asset treasuries. A weight change for this pool would be instantly reflected. The pool will rapidly adjust through arbitrage to the intended value design by offering stable liquidity in either direction [88]. A drawback of this design is the larger price impact associated with significant weight changes.

5.2.2 Auction based re-balancing

To mitigate the price impact of large re-balances, Set Protocol [41] uses a Dutch auction mechanism for discrete re-balances where individual auctions can be further split into chunks to target TWAP. The auction mechanism achieves more attractive pricing by delaying the re-balance.

There are two challenges with the auction model that makes it less suitable for the treasury re-insurance use case:

- Auctions are not amenable to frequent interruption. If there are several consecutive withdrawal events or parameter submissions, auctions would have to be repeatedly restarted possibly front-running re-balance trades
- Auctioned pools provide only intermittent liquidity. Predictable and stable liquidity attracts more trade volume (including more organic volume) which would contribute to lower re-balancing costs for Aera vaults in the long run. We think the auction model could be introduced with time with sufficient evidence of auction liquidity

5.2.3 Limitations of gradual re-balancing

The goal of the Aera vault gradual re-balancing scheme is to combine the ability of Balancer pools to attract trade volume with auction-like mechanics to achieve price execution that is closer to TWAP or VWAP.

However, it is unclear if the gradual Balancer weight adjustment provides the optimal price transition curve. It's also not easy for traders with large trades to provide liquidity in this model.

5.2.4 Alternatives

We have also considered a hold-out model that temporarily removes from trading (and lends) and/or reintroduces portions of tokens in the event of a withdrawal or parameter change. These tokens could then be drip-fed back into the Balancer pool allowing for alternative price curves. We expect further simulation work and on-chain testing to inform future refinements.

6 Contract Specifications (V1)

6.1 Withdrawal validator

The withdrawal policy is expressed on-chain with a withdrawal validator contract. A withdrawal validator contract will be deployed and will be able to recognize shortfall events that could trigger withdrawals. The withdrawal validator is agreed between the guardian and the treasury owner and could be deployed by either. Its interface is a single function:

- `allowance` returns a list of `uint` values with the corresponding token amounts.

6.2 Aera vault

The V1 Aera Vault contract supports the following functions:

- `initialDeposit`, needs to be called by the owner to provide initial funding for the Aera vault

- `deposit`, allows owner to make additional deposits at any time while preserving existing pool spot price. Should generally not be called if trading is enabled
- `depositIfBalanceUnchanged`, a deposit operation that checks if the underlying Balancer pool balance hasn't changed in the current block as a form of simple MEV-protection
- `withdraw`, request by the owner to return a given amount of each token back to the protocol treasury (owner) while preserving existing pool spot price. Should generally not be called if trading is enabled
- `withdrawIfBalanceUnchanged`, similar to `depositIfBalanceUnchanged`
- `enableTradingRiskingArbitrage`, enable trading with current spot prices between pool assets. Vulnerable to arbitrage if spot prices do not agree with current market prices
- `enableTradingWithWeights`, enable trading with a specific set of weights to control spot prices and limit arbitrage
- `initiateFinalization`, request by the owner or guardian to initiate notice period for vault termination
- `finalize`, confirms vault finalization after notice period has expired
- `setManager`, allows owner to change guardian
- `sweep`, allows owner to recover tokens sent in error
- `claimRewards`, claim underlying Balancer pool rewards
- `disableTrading`, disable trading in an emergency or prior to a deposit/withdraw action
- `updateWeightsGradually`, controls the target Balancer weight by specifying the target weights of tokens, start and end time of weight transition schedule
- `cancelWeightUpdates`, cancels active weight update
- `setSwapFee`, allows guardian to change current swap fee
- `claimManagerFees`, allows guardian to claim pool fees if enabled and earned

6.3 Security

The V1 implementation of Aera vaults has been independently audited by Spearbit. The full report is published here.

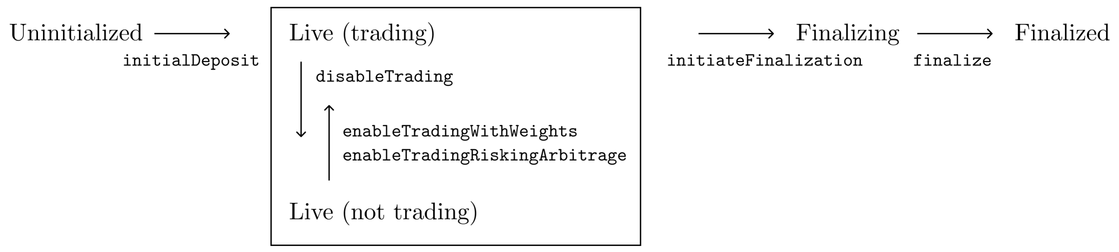

Figure 5: State transition model for a single vault

State	**Uninitialized**	**Trading**	**Not trading**	**Finalizing**	**Finalized**
Owner					
`initialDeposit`	x				
`deposit`		x (rarely)	x		
`depositIfBalanceUnchanged`		x (rarely)	x		
`withdraw`		x (rarely)	x		
`withdrawIfBalanceUnchanged`		x (rarely)	x		
`enableTradingRiskingArbitrage`		x (no effect)	x (rarely)	x (rarely)	
`enableTradingWithWeights`		x (no effect)	x	x (rarely)	
`initiateFinalization`		x	x		
`finalize`				x	
`setManager`	x	x	x	x	x
`sweep`	x	x	x	x	x
`claimRewards`		x	x	x	
Emergency (owner or guardian)					
`disableTrading`		x	x (no effect)	x (rarely)	
Guardian					
`updateWeightsGradually`		x	x		
`cancelWeightUpdates`		x	x		
`setSwapFee`		x	x	x	
`claimManagerFees`		x	x		

Table 3: Lifecycle model for a single vault

7 Simulation

Background on Agent-Based Simulation The main tool that we use in our analysis of Aera protocol is agent-based simulation (ABS). ABS has been used in a variety of contexts in quantitative finance, including to estimate censorship in cryptocurrency protocols [34], detect fraudulent trading activity in CFTC exchanges [138], stress test frameworks from the European Central Bank [59, 83] and the Federal Reserve [51, 21] and find optimal fee size geometric-mean automated market makers such as Balancer [39]. These models can provide invaluable information on the behavior of complex systems. This has made ABS widely used in industries such as algorithmic trading and self-driving car deployment.

Gauntlet Simulation Environment The Gauntlet platform, which was used for all simulations and results in this paper, provides a modular, generic ABS interface for running simulations directly against Proof of Stake consensus mechanisms [29] and Ethereum smart contracts [72]. In this system, the agent models are specified via a Python domain-specific language (DSL), akin to Facebook's PyTorch [107], and interact with blockchain components in their native languages. Agents can also interact with non-blockchain modules, such as historical or synthetic market data and/or other off-chain systems. The DSL hides the blockchain-level details from the analyst, allowing the end-user to develop strategies that can migrate from one smart contract to another, should they have similar interfaces. Most of the platform's design is inspired by similar platforms in algorithmic trading that allow for quantitative researchers to develop strategies that execute over multiple exchanges (with varying order books, wire protocols, slippage models, etc.) without having to know these low-level details. Moreover, the non-blockchain portions of the simulation are analogous to trading back-testing environments [104], so that agents are interacting with realistic order books and financial data.

7.1 Simulation Setup

7.1.1 Contracts

Lending Protocol To support the analysis of the impact of the Aera vault for multiple lending protocols we utilize a generic lending contract with loan books initialized based on the empirical distributions of each protocol as seen in the example given in Figure 6. For any individual simulation run there is a lending contract agents interact with for loans and liquidations that corresponds to a specific protocol of interest. We ran over a million simulations across the Aave, Compound and Benqi protocols.

Each loan book is largely split into two components which we define as long risk referencing volatile asset collateralized stable loans and short risk referring to stable asset collateralized volatile asset loans. This is described in more detail in E.2.

Exchange In the simulation trades occur on an internal representation of an external exchange that allows agents to buy and sell assets. It also acts as the fair price oracle

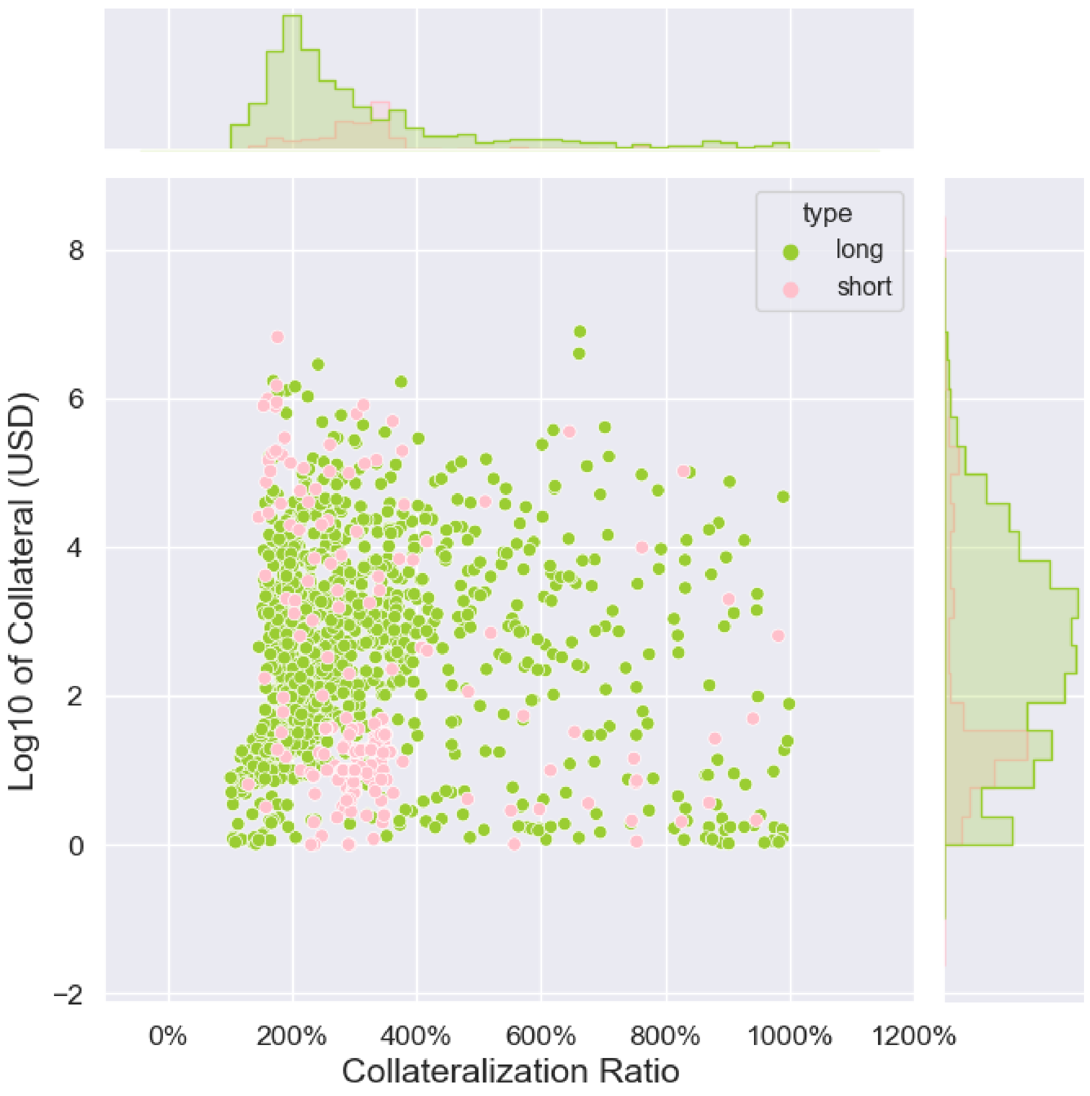

Figure 6: Benqi loan distribution

for the other agents and contracts in the sim. We define slippage models trained off of historical liquidity data to accurately represent the price impact of trades in the simulation environment where larger trade sizes incur more slippage.

More specifically, the slippage s is defined as

$$s = I\sigma \min\left(\frac{x}{\text{ADV}}, 1\right)^{\beta}$$

where x is the order size, ADV denotes the average daily traded volume, I is a scalar for intensity, σ is the annualized volatility and β controls the concavity of the curve. This model is derived from a family of models popularly used both in traditional financial literature [6] and in practice by risk modelers [133].

Vault Each simulation has a vault that corresponds to a treasury allocation. The space of vaults includes alternatives such as 80/20 Balancer pools or simply a pure one time diversification into stables which are simulated to provide a performance comparison of the Aera vault against other solutions.

As prices move around naturally there are arbitrageurs that trade the spot price of the vault and the fair price on exchange back in line. The Aera vault has a rebalancing strategy that continuously updates weights in accordance to the composition of the lending protocol it is tracking. This is described in more detail in Appendix E.

7.1.2 Agents

Borrower Borrowers are initialized with outstanding loans sampled from representative distributions modeled off of protocol state with an example shown in Figure 7. Over the course of the simulation borrowers periodically update their position to maintain a target collateralization ratio. If a borrower's loan becomes insolvent it is assumed they will keep the borrowed assets and leave their collateral behind for a profit.

Liquidator In the simulation, liquidators act as profit seeking agents that continuously monitor borrower positions for undercollateralization. If at any point a loan's collateralization ratio dips below the lending protocol's liquidation threshold a liquidator steps in to reduce the overall asset exposure of the loan to the protocol by purchasing collateral through debt repayment for a discount to the exchange rate.

When opportunities arise, liquidator agents execute the most profitable liquidation factoring in costs such as slippage incurred in rotating out of the collateral asset, exchange trade fees, gas costs etc and will only execute the transaction if the expected profit from the liquidation bonus exceeds these costs. Extreme scenarios where liquidations don't occur due to increased costs, sharp price movements etc can be catastrophic for the protocol and lead to systemic insolvencies. There are also cases where accounts become insolvent as a result of liquidation cascades that occur due to the cumulative price impact of successive liquidations triggering more liquidations which is similarly detrimental to protocol health and thus closely tracked in simulation.

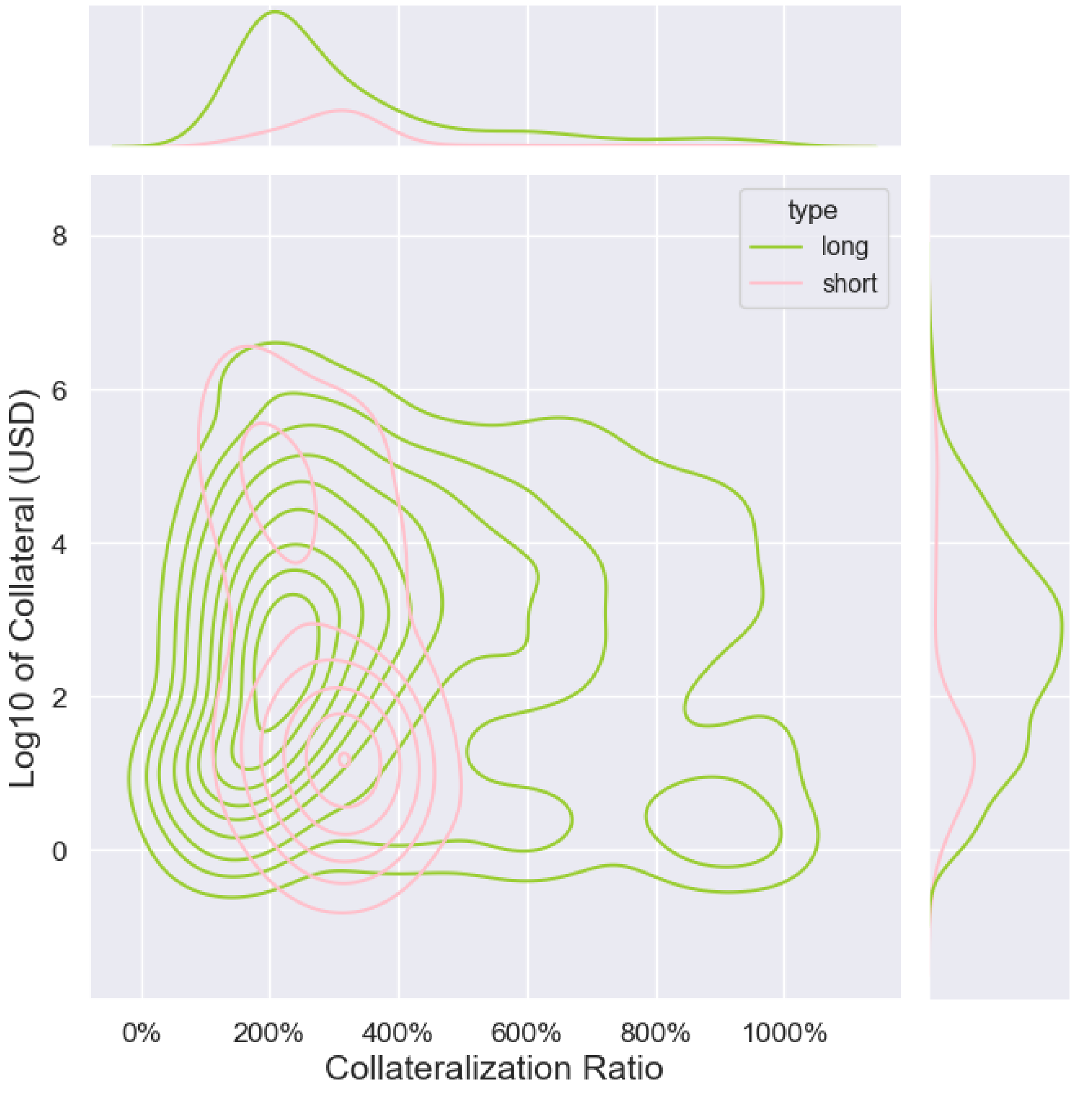

Figure 7: Benqi loan distribution kernel density estimate

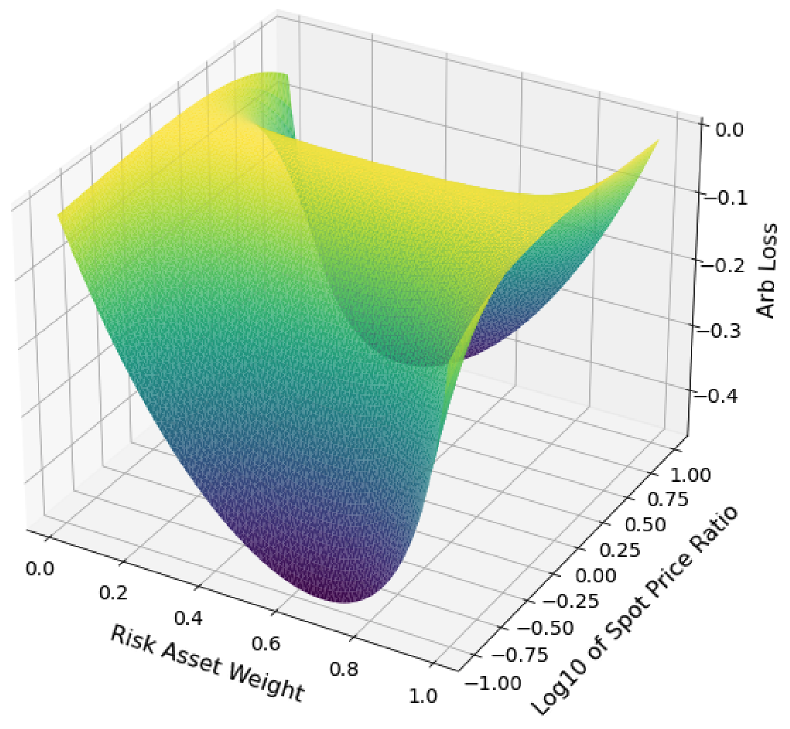

Figure 8: Arb loss from price changes

Arbitrageur Arbitrageurs are profit seeking agents that continuously monitor the prices of liquidity venues. If at any point the asset prices on one venue diverge sufficiently from another such that a opportunity exists where an actor can enter an opposing trade on each venue that results in riskless profit the execution of such trades is referred to as an arbitrage or an arb for short.

The scale of price dislocation that can occur is dependent on the costs associated with an arb execution strategy. In the simulation arb agents trade between the vault and the exchange and incur slippage and fee costs on both venues. It is assumed that arb agents execute the profit maximizing trades immediately upon arrival and that trades are only done if the sequence is profitable. The gains to the arb agent and the marked to market profit of the exchange at the time of each trade correspond in a zero sum fashion to the arb loss for the vault where the vault is functionally compensating arb agents for the service of moving spot prices in line with the fair oracle price from the exchange.

The magnitude of the arb can depend on a variety of factors such as the ratio of the vault spot price to the fair price on exchange and the weights of the underlying asset pool. This relationship can be seen in the surface and cross sectional plots in Figures 8 and 9.

Parameter Submitter After each epoch period, the parameter submitter computes all the positional deltas of the loan book and combines the long risk as shown in the example in Figure 10 with the short risk as shown in the example in Figure 11 which are the primary

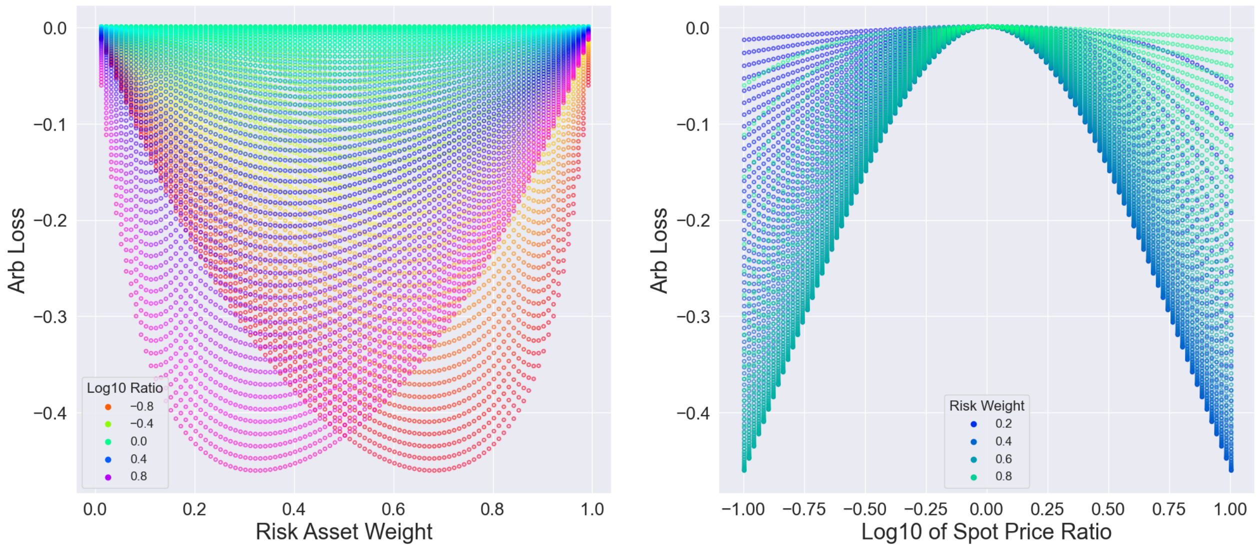

Figure 9: Arb loss from price changes cross sections

inputs into the net position value of the loan book which is detailed in E.2.

The aggregated risk value is then passed into the construction of a target portfolio, a process described in more detail in E.3. From there the submitter builds an update schedule that consists of the target portfolio weights and a rebalance window over which the weight change will occur and subsequently applied to the Aera vault.

7.2 Simulation Parameters

Since the parameter space in this setup is quite high dimensional, in order to simplify the analysis of the tradeoff surface parameters external to the vault are grouped into distinct clusters which are labeled and each referred to as a distinct market context.

7.2.1 Market context

Price trajectories One of the prominent components within a market context is the price trajectory which is produced by sampling a stochastic process. A common model for standard market conditions is given by multivariate geometric Brownian motion where the price process S_t obeys the Itô stochastic differential equation (SDE) given by

$$dS_t^i = \mu_i S_t^i dt + \sigma_t^i S_t^i dW_t^i$$

where W_t, μ and σ denote the Wiener process, annualized drift and volatility respectively for asset i and ρ denotes the correlation matrix of log returns such that the Wiener processes are correlated so

$$\mathbf{E}\left[dW_t^i dW_t^j\right] = \rho_{i,j}$$

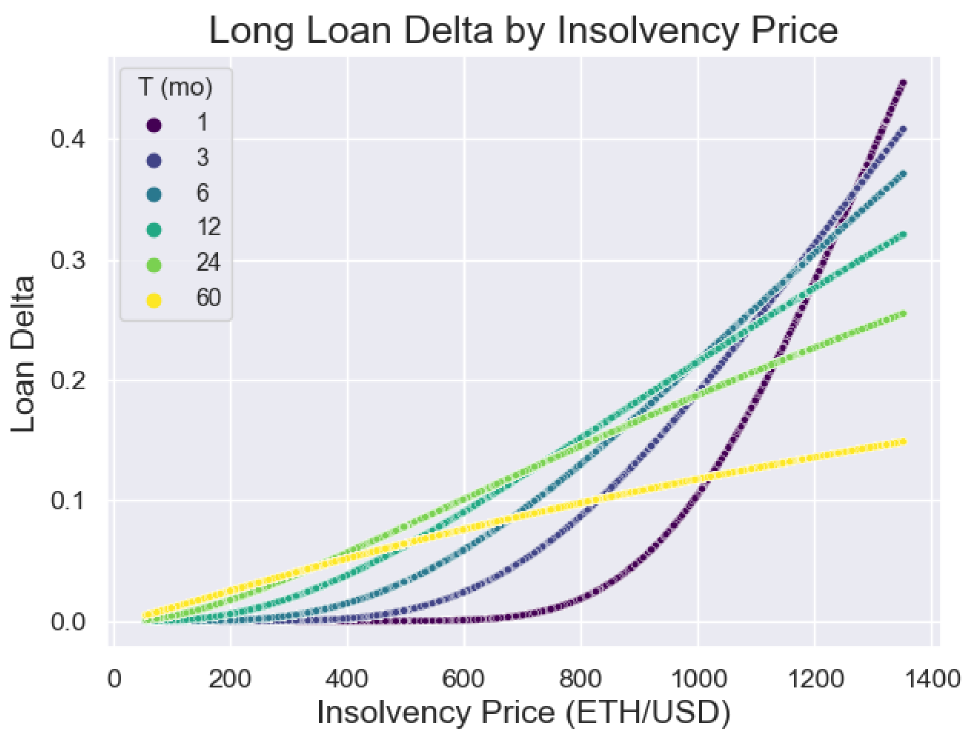

Figure 10: Compound long loan deltas

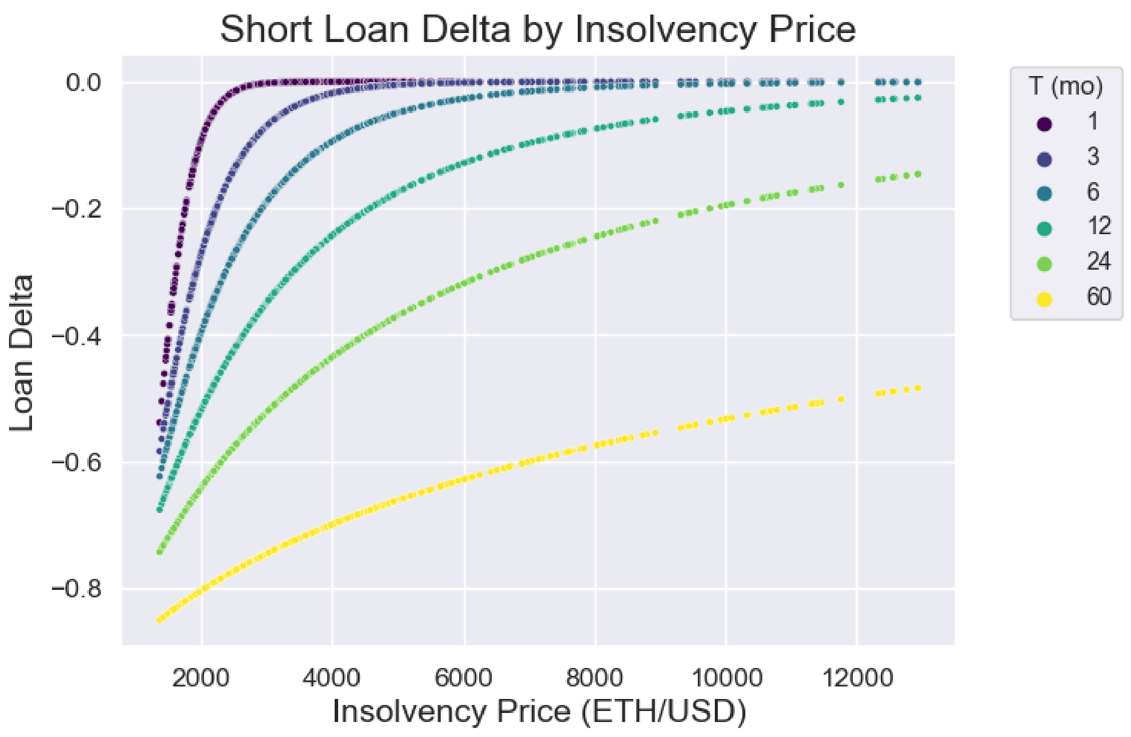

Figure 11: Compound short loan deltas

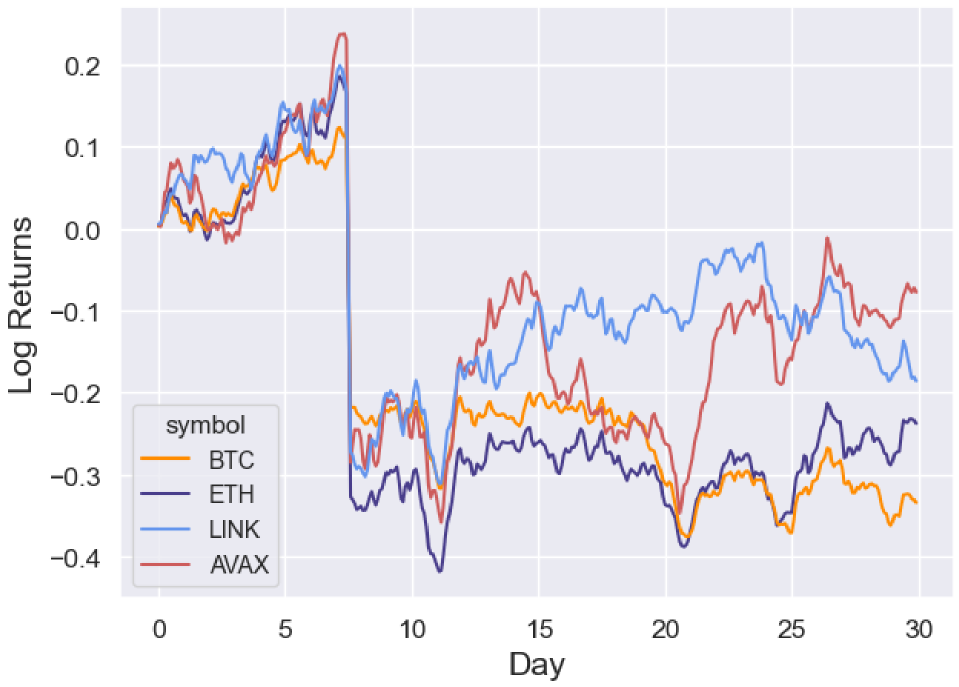

Figure 12: Crash price trajectory generated with multivariate geometric Brownian motion with jumps

One particular trajectory of interest is that of a market crash like Black Thursday for MakerDAO or the 2010 U.S. flash crash that applies substantial financial stress to a protocol. In order to model this we compose the base process above with the Merton jump-diffusion model which is a jump process composed of log normal jumps given by a Poisson process and a diffusion process governed by geometric Brownian motion. This model allows us to simulate abrupt price changes that triggers heightened rates of liquidations for the protocol while also being analytically tractable and the standard model used by the Federal Reserve for defaultable securities [141].

Formally the jump diffusion price process S_t obeys the SDE

$$\frac{dS_t}{S_t} = (\mu - \lambda\mu_k)dt + \sigma dW_t + k dX_t \tag{8}$$

where W_t is the Wiener process, μ and σ denote the percentage drift and volatility, X_t is a compound Poisson process with intensity λ and jump magnitude k which is parameterized by a log normal with mean γ and variance δ^2 in

$$\ln(1 + k) \sim \mathcal{N}(\gamma, \delta^2) \tag{9}$$

and $\mu_k = e^{\gamma+\delta^2/2} - 1$. An example of this is shown in Figure 12.

Another trajectory of interest is that of a prolonged bear market that could test the insolvency coverage of a protocol. One approach to model this is to compose the base

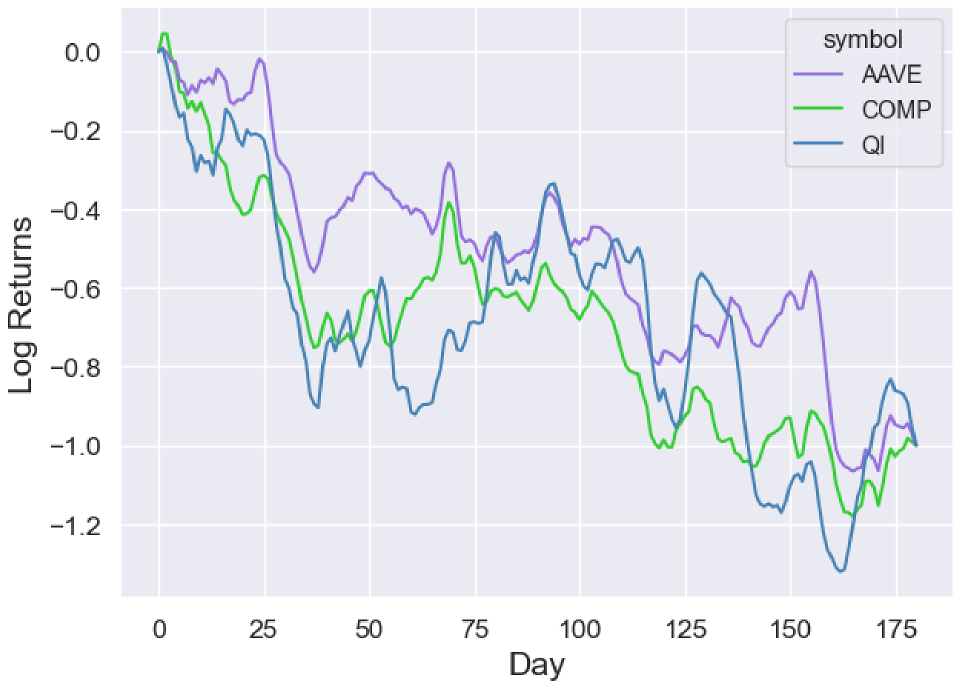

Figure 13: Bear price trajectory generated with multivariate geometric Brownian bridge

process above with a Brownian bridge, a process which is similar to Brownian motion except the endpoints are pinned to predetermined values. This is helpful for simulating trajectories with large drawdowns over time such as the 50% drop in the S&P 500 Index from Sept 2008 to Feb 2009 or the 75% drop in ETH from Nov 2021 to Jun 2022.

The Brownian bridge process S_t is defined by

$$S_t = \frac{T-t}{T} \cdot a + \frac{t}{T} \cdot (b - W_T) + W_t$$

where $t \in [0, T]$ for some end time T, W_t is the standard Wiener process with boundary conditions $S_0 = a$ and $S_T = b$. An example of this is shown in Figure 13.

Besides normal, crash and bear market contexts we also generate trajectories for bull and volatile markets which correspond to high drift and high volatility parameterizations of the base process.

Adversarial behavior Periods of financial stress for the protocol tend to create opportune moments for profit seeking economic actors to collect a large windfall, often at the expense of the protocol. We consider agent behavior that may materialize in extreme market conditions such as the removal of liquidity and increase in slippage in periods of high volatility, adversarial large borrows given the subsequently increased costs for liquidation and intense short selling with the expressed purpose of triggering cascading liquidations for profit.

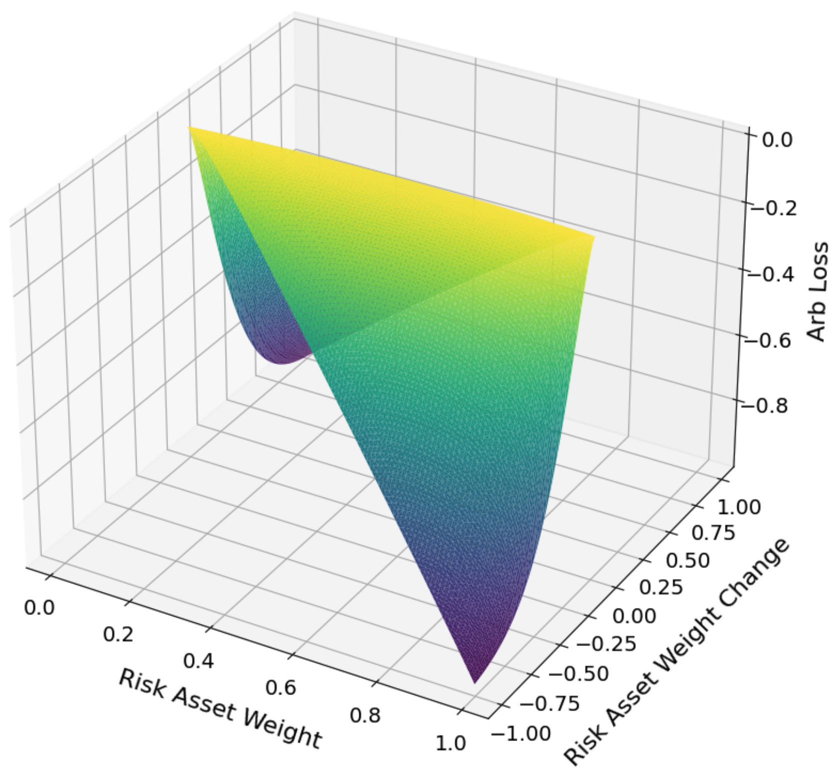

Figure 14: Arb loss from weight changes

7.2.2 Vault experiment

We simulate Aera vaults under a wide swath of configurations for the Aave, Compound and Benqi protocols. From a three asset system composed of a risk asset, a stable asset and a governance asset we also construct comparison vaults, namely the 80/20 Balancer pool vault for each pair of assets and vaults containing each singular asset.

Vault design We trialed a variety of Aera vault designs with features such as oracle based pricing, asymmetric fees, dynamic slippage etc and ultimately decided on the current design based on experiment performance and Solidity development time.

Vault strategy The Aera vault's primary objective is to achieve risk reduction through dynamic asset reallocation with low execution costs. An important consideration of each rebalance is the associated cost in the form of arb loss. The weight change arb loss surface is quite different from the price change one as seen in the surface and cross section plots in Figures 14 and 15.

A noticeable difference is that price changes are more costly when weights are more similar while weight changes are most expensive when the weights are farthest apart. For price changes there is a tradeoff with both slippage and opportunity size increasing towards

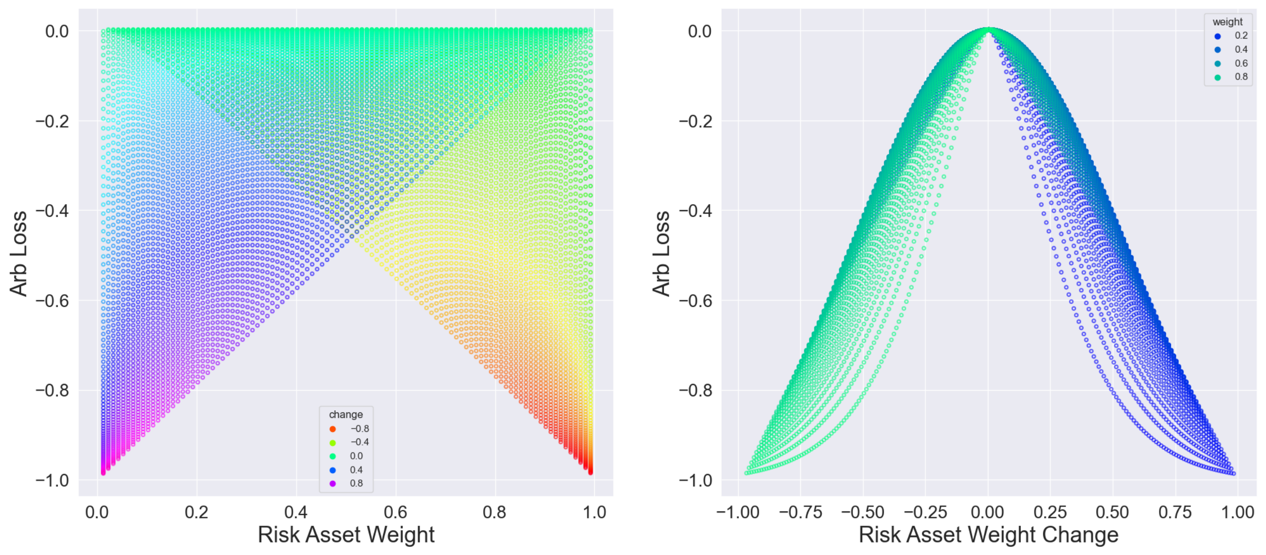

Figure 15: Arb loss from weight changes cross sections

uneven weights but for weight changes the magnitude of spot price movement increases dramatically with weight imbalance.

This becomes an important consideration for the parameter submitter in deciding the vault rebalance strategy especially for tuning update speed parameters which are addressed in more detail in E.3. Unlike price changes whose opportunity are path independent weight change arb losses are exacerbated the longer spot price dislocations persist. As such we run backstop arb agents for each Aera vault to ensure that weight changes occur in a timely and cost effective manner.

7.3 Simulation Results

7.3.1 Objective Function

Referencing the results in Figure 16 we see objective function performance as defined in E.4 for the Aera vault and note it outperforms comparison vaults collectively across the various market contexts by being the only vault able to generate returns while mitigating downside. These results are stable across simulations for Aave, Compound and Benqi.

In comparison to a vault containing only stables Aera unsurprisingly does worse in highly negative drift markets as shown in the bear market context but performs better in markets with positive drift both in terms of APY and in the presence of heightened volatility has the ability to cover large loan book short risks that stablecoins do not. Since risk tends to be relatively low in a bull market the Aera vault rotates mostly into risk assets which allows it to do better than vaults with a 20% stable composition while achieving competitive returns with vaults containing only volatile assets.

Due to the heightened volatility in crash and volatile markets, the Aera vault acts to dynamically rotate in stable assets and while this rebalance cost is nontrivial it pays off

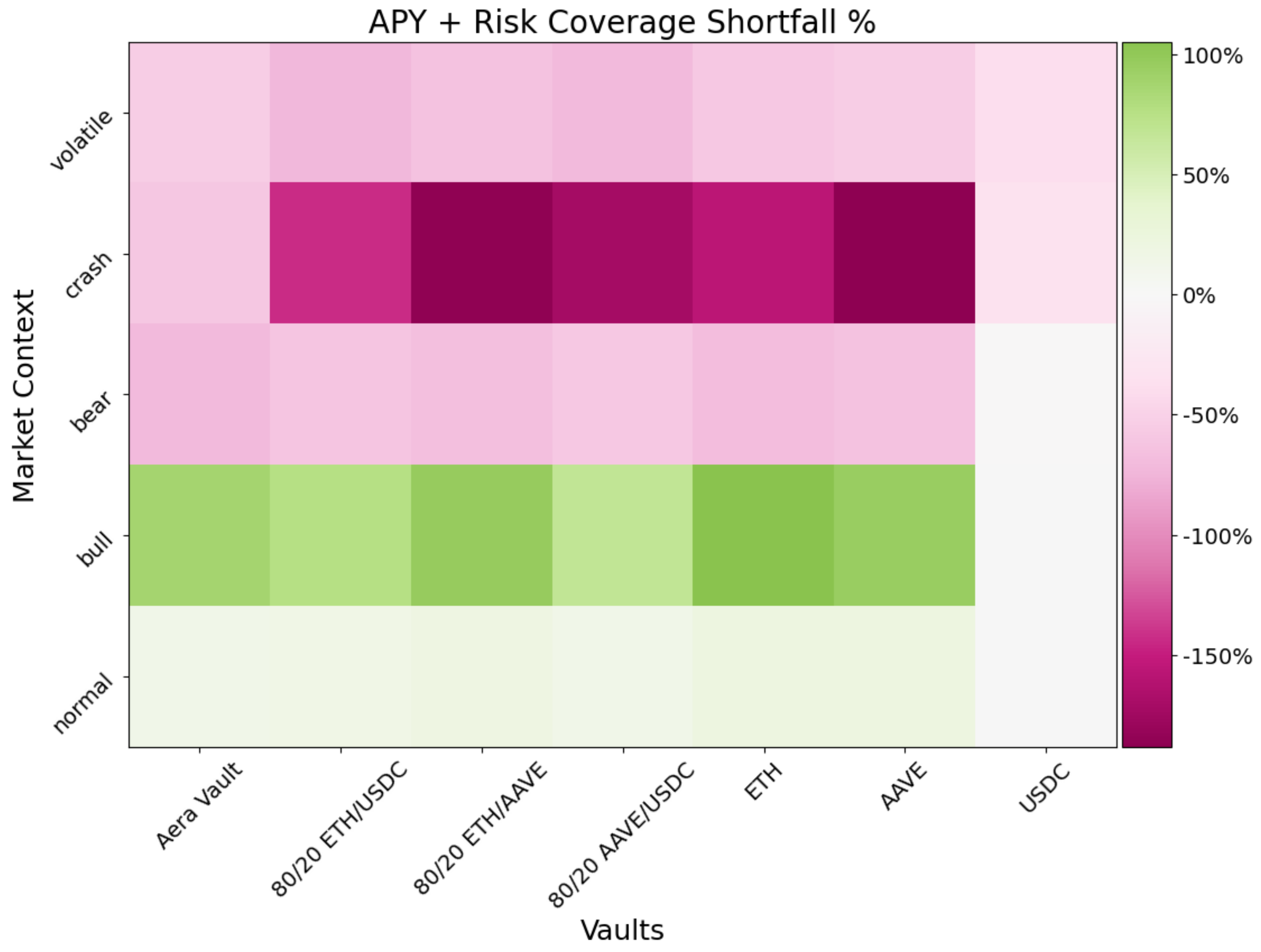

Figure 16: Aera Aave vault objective function performance

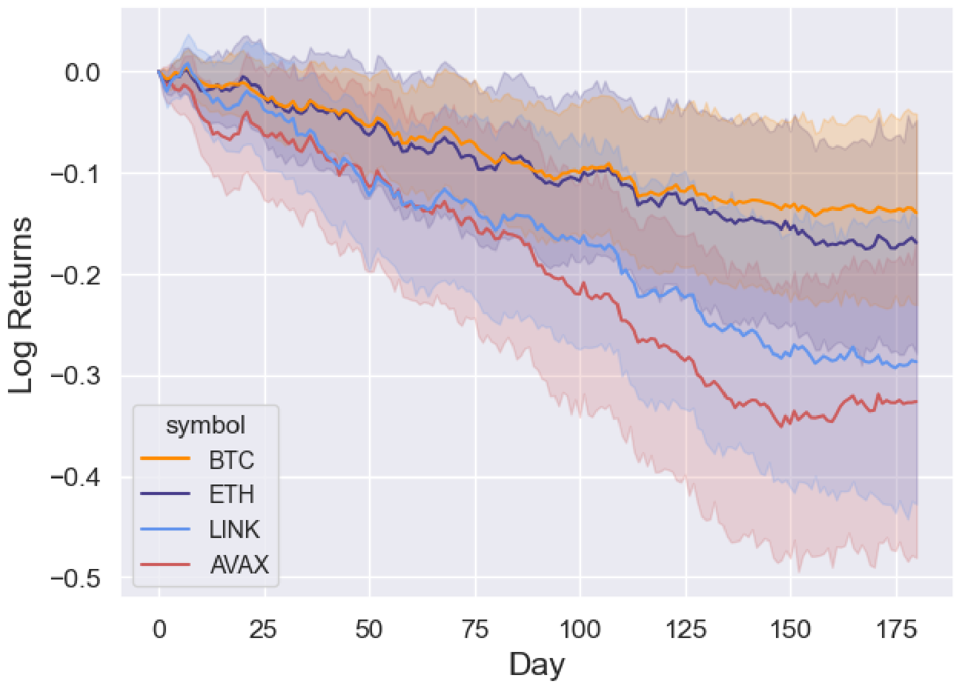

Figure 17: Volatility drag in simulated multivariate geometric Brownian motion prices

significantly in a crash market as evidenced in the results. The difference in performance between vaults containing purely risk assets versus governance assets can in part be explained by liquidity differences and also by the volatility drag from substantially higher vol for governance assets with comparable drift as shown in Figure 17. This is another consideration that can be optimized for with dynamic rebalancing with Aera.

7.3.2 Rebalancing Costs

Tracing the arb loss in Figure 18 we see the total rebalance costs incurred to the protocol are smaller than 80/20 counterparts with a stable asset despite having the additional cost of weight changes. This follows from extensive parameter tuning and backtesting of the vault rebalance strategy along with detailed analysis of the arb loss surface. The Aera vault has comparable arb loss to that of the 80/20 vault with a risk and governance asset which is reduced from the strong correlation between the two assets but also carries none of the benefits of having a stable asset.

8 Future Work

While sections 5 and 6 focus on the initial implementation of Aera, there are a number of features being developed for future versions of Aera.

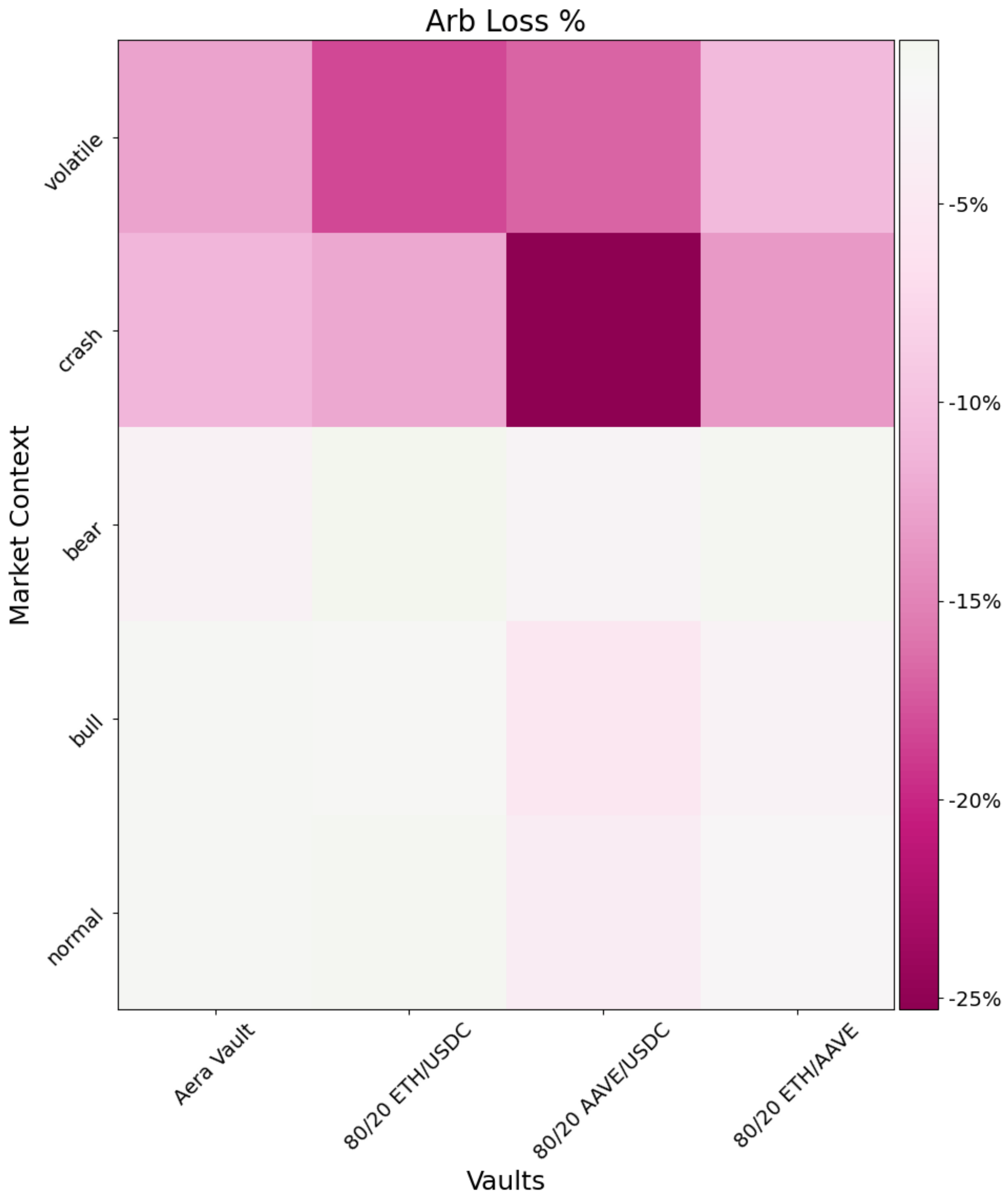

Figure 18: Aera Aave vault arbitrage loss

8.1 Aera V2 contracts

The next iteration of Aera contracts will be V2 and will resolve several shortcomings of the V1 contract implementation.

Oracle prices for trade enablement and deposits. In order to prevent front-running and arbitrage in the current vault, it is strongly recommended to enable trading with market-aware spot prices. Furthermore, it is recommended to disable trading when performing a deposit. For some treasuries, it won't be convenient to agree on and provide market-aware spot prices in a timely manner. Instead, V2 will rely on an external price oracle and introduce two new functions: one to enable trading with oracle prices and another to enable deposits informed by oracle prices. The deposit function in particular is able to combine information provided by the active underlying Balancer pool spot price and the oracle price for increased safety.

Multicall. V2 will support multicall, allowing treasuries to take several actions atomically. For example, this could allow a treasury to disable trading, withdraw funds and re-enable trading with an oracle price atomically.

Removal of notice period. The notice period is a cumbersome way to align incentives around long-term management between the treasury and parameter submitters. Instead, finalization will be immediately available but there will be a minimum management period for which parameter submitters will be able to earn and collect fees.

Generalized asset support. While the current V1 vault only supports ERC20 assets, the V2 vault will be extended to support ERC4626 assets. In turn, this will allow the Aera vaults to allocate capital into yield-bearing assets, staking opportunities as well as execute more complex execution flows. The use of ERC4626 assets will have some restrictions, for example, the underlying asset will need to exist as an ERC20 asset in the Aera vault to facilitate rebalancing.

Crash put purchasing. Further to the above, V2 will allow Aera vaults to purchase out of the money put options to hedge against downward market movements in risk assets. The purchasing flow will be abstracted in a dedicated crash put purchasing vault. This will allow us to improve the optimization and execution quality for the problems specified in 4.1.

8.2 Aera V3 contracts

The main milestone for the third version of the Aera contracts is to include the full staking system as described in 3 and 4. Constructing a computationally efficient and economically secure implementation of the grading and hedging functionalities is crucial to ensuring that Aera can be used by virtually all DAOs in DeFi for many different objective functions.

Multiple parameter submitters. V3 will allow a (whitelisted) set of multiple parameter submitters to control a single vault using an aggregation and grading policy. Slashing will not yet be applied in V3.

Fixed epoch. In order to facilitate multiple parameter submitters, the epoch schedule and frequency will be fixed.

On-chain aggregation. Parameter submissions will be aggregated on-chain using a medianizer approach.

Grading oracle. V3 will use an oracle to compute Shapley grades off-chain (given the complexity of the risk objective function) and submit them to the vault.

Yield bearing assets. It is expected that we may add yield-bearing assets to the asset universe, allowing the vault to invest idle capital. Yield-bearing assets will be introduced using the ERC4626 interface.

9 Authors

For provenance, the authors of this paper are: Tarun Chitra, Peteris Erins, Victor Xu, Henry Prior, Nicholas Borg, Shaan Varia, Hsien-Tang Kao, Rei Chiang, John Morrow, and Kemen Linsuain.

10 Acknowledgments

The authors would like to acknowledge Kshitij Kulkarni, Rahul Goyal, Zachary Fallon, Elena Burger, Adam Lerer, Lily Francus, James Hutchings, Oliver Bell, Alok Vasudev for helpful feedback.

References

[1] 0xMaki. 2021. [withdrawn] sushi phantom troupe - strategic raise. `https://forum.sushi.com/t/withdrawn-sushi-phantom-troupe-strategic-raise/4554`

[2] 0xngmi. 2022. DeFi Llama. `https://defillama.com/`

[3] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. 2021. Uniswap v3 Core. (2021).

[4] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. 2018. Astraea: A decentralized blockchain oracle. In *2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 1145–1152.

[5] Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. 2019. Online control with adversarial disturbances. In *International Conference on Machine Learning*. PMLR, 111–119.

[6] Robert Almgren, Emmanuel Hauptmann, Jong Li, and Chee Thum. 2005. Direct Estimation of Equity Market Impact. *Risk* 18 (2005), 57–62.

[7] Noga Alon, Nicolo Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. 2017. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM J. Comput.* 46, 6 (2017), 1785–1826.

[8] Guillermo Angeris, Akshay Agrawal, Alex Evans, Tarun Chitra, and Stephen Boyd. 2021. Constant function market makers: Multi-asset trades via convex optimization. *arXiv preprint arXiv:2107.12484* (2021).

[9] Guillermo Angeris and Tarun Chitra. 2020. Improved Price Oracles: Constant Function Market Makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. ACM, New York NY USA, 80–91. `https://doi.org/10.1145/3419614.3423251`

[10] Guillermo Angeris, Alex Evans, and Tarun Chitra. 2020. When does the tail wag the dog? Curvature and market making. *arXiv preprint arXiv:2012.08040* (2020).

[11] Guillermo Angeris, Alex Evans, and Tarun Chitra. 2021. Replicating Market Makers. *arXiv preprint arXiv:2103.14769* (2021).

[12] Anzen. [n.d.]. Anzen Overview. `https://docs.anzen.finance/overview/`

[13] Pyth Data Association. 2022. `https://pyth.network/whitepaper.pdf`

[14] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th annual foundations of computer science*. IEEE, 322–331.

[15] Marco Avellaneda and Jeong-Hyun Lee. 2010. Statistical arbitrage in the US equities market. *Quantitative Finance* 10, 7 (2010), 761–782.

[16] Marco Avellaneda and Sasha Stoikov. 2008. High-frequency trading in a limit order book. *Quantitative Finance* 8, 3 (2008), 217–224.

[17] Andrey Belyakov. 2020. First Credit default swap on AAVE credit delegation launched. https://medium.com/opium-network/first-credit-default-swap-on-aave-credit-delegation-launched-5e3efc961317

[18] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).

[19] Ernesto Boado. 2022. Aave DAO Treasury diversification discussion. https://governance.aave.com/t/aave-dao-treasury-diversification-discussion/7961

[20] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[21] Richard Bookstaber, Mark Paddrik, and Brian Tivnan. 2018. An agent-based model for financial vulnerability. *Journal of Economic Interaction and Coordination* 13, 2 (2018), 433–466.

[22] Stephen Boyd, Enzo Busseti, Steve Diamond, Ronald N Kahn, Kwangmoo Koh, Peter Nystrup, Jan Speth, et al. 2017. Multi-period trading via convex optimization. *Foundations and Trends® in Optimization* 3, 1 (2017), 1–76.

[23] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. 2021. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs* (2021).

[24] Vitalik Buterin. 2014. An introduction to Futarchy. *Ethereum Blog* (2014).

[25] David Canellis. 2022. Uniswap hits $1T trade volume milestone. https://blockworks.co/uniswap-hits-1t-trade-volume-milestone/

[26] Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. 2007. Improved second-order bounds for prediction with expert advice. *Machine Learning* 66, 2 (2007), 321–352.

[27] Chandler. 2021. Introducing milestone KPI options. https://medium.com/uma-project/introducing-milestone-kpi-options-2a8ea4cf480e

[28] Yiling Chen, Lance Fortnow, Nicolas Lambert, David M Pennock, and Jennifer Wortman. 2008. Complexity of combinatorial market makers. In *Proceedings of the 9th ACM Conference on Electronic Commerce*. 190–199.

[29] Tarun Chitra. 2020. Competitive equilibria between staking and on-chain lending. *arXiv preprint arXiv:2001.00919* (2020).

[30] Tarun Chitra. 2022. Building and running a DAO: Why governance matters. https://future.com/building-and-running-a-dao-why-governance-matters/

[31] Tarun Chitra, Guillermo Angeris, and Alex Evans. 2021. How Liveness Separates CFMMs and Order Books. (2021). https://stanford.edu/~guillean/papers/cfmm-ob.pdf

[32] Tarun Chitra, Guillermo Angeris, Alex Evans, and Hsien-Tang Kao. 2021. A Note on Borrowing Constant Function Market Maker Shares. (2021). https://stanford.edu/~guillean/papers/cfmm-lending.pdf.

[33] Tarun Chitra and Alex Evans. 2020. Why stake when you can borrow? *arXiv preprint arXiv:2006.11156* (2020).

[34] T. Chitra, M. Quaintance, S. Haber, and W. Martino. 2019. Agent-Based Simulations of Blockchain protocols illustrated via Kadena's Chainweb. (June 2019), 386–395. https://doi.org/10.1109/EuroSPW.2019.00049

[35] Richard Craib, Geoffrey Bradway, Xander Dunn, and Joey Krug. 2017. Numeraire: A cryptographic token for coordinating machine intelligence and preventing overfitting. *Retrieved* 23 (2017), 2018.

[36] Joachim De Lataillade, Cyril Deremble, Marc Potters, and Jean-Philippe Bouchaud. 2012. Optimal trading with linear costs. *arXiv preprint arXiv:1203.5957* (2012).

[37] Steven De Rooij, Tim Van Erven, Peter D Grünwald, and Wouter M Koolen. 2014. Follow the leader if you can, hedge if you must. *The Journal of Machine Learning Research* 15, 1 (2014), 1281–1316.

[38] Payom Dousti. 2022. What is cozy? https://docs.cozy.finance/

[39] Alex Evans, Guillermo Angeris, and Tarun Chitra. 2021. Optimal fees for geometric mean market makers. *arXiv preprint arXiv:2104.00446* (2021).

[40] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*. 1605–1622.

[41] Brian Weickmann Felix Feng. 2019. Set: A Protocol for Baskets of Tokenized Assets. (2019). https://www.setprotocol.com/pdf/set_protocol_whitepaper.pdf.

[42] Matheus VX Ferreira and S Matthew Weinberg. 2020. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In *Proceedings of the 21st ACM Conference on Economics and Computation*. 683–712.

[43] Arrakis Finance. 2022. Introducing Arrakis - Web3's liquidity layer. https://medium.com/arrakis-finance/introducing-arrakis-web3s-liquidity-layer-ae656dd411

[44] Ribbon Finance. 2021. Ribbon Treasury. `https://ribbonfinance.medium.com/ribbon-treasury-ee311f7ce7d8`

[45] Bank for International Settlements. 2022. OTC derivatives statistics at end-December 2021. `https://www.bis.org/publ/otc_hy2205.htm`

[46] Gunnar Friede, Timo Busch, and Alexander Bassen. 2015. ESG and financial performance: aggregated evidence from more than 2000 empirical studies. *Journal of sustainable finance & investment* 5, 4 (2015), 210–233.

[47] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464* (2019).

[48] Peter M Garber. 1998. Derivatives in international capital flows.

[49] Alexis Gauba. [n.d.]. Opyn V1 GitBook. `https://opyn.gitbook.io/opynv1/faq`

[50] Alexis Gauba. [n.d.]. Opyn v2 GitBook. `https://opyn.gitbook.io/opyn/`

[51] John Geanakoplos, Robert Axtell, J Doyne Farmer, Peter Howitt, Benjamin Conlee, Jonathan Goldstein, Matthew Hendrey, Nathan M Palmer, and Chun-Yi Yang. 2012. Getting at systemic risk via an agent-based model of the housing market. *American Economic Review* 102, 3 (2012), 53–58.

[52] William George and Clément Lesaege. [n.d.]. An Analysis of p+ ε Attacks on Various Models of Schelling Game Based Systems. ([n. d.]). `https://assets.pubpub.org/33am50c6/51581340370715.pdf`.

[53] Guido Giese, Linda-Eling Lee, Dimitris Melas, Zoltán Nagy, and Laura Nishikawa. 2019. Foundations of ESG investing: How ESG affects equity valuation, risk, and performance. *The Journal of Portfolio Management* 45, 5 (2019), 69–83.

[54] Mark E Glickman. 1995. The glicko system. *Boston University* 16 (1995), 16–17.

[55] Gnosis. 2021. Introducing gnosis auction. `https://medium.com/@gnosisPM/introducing-gnosis-auction-ac30232b3595`

[56] Anand Gomes. [n.d.]. Paradigm announces the first DOV partnership with Ribbon Finance: Partnerships. `https://www.paradigm.co/blog/paradigm-announces-the-first-dov-partnership-with-ribbon-finance`

[57] govtrack.us. 2021. Historical statistics about legislation in the U.S. congress. `https://www.govtrack.us/congress/bills/statistics`

[58] Hannes Graah. 2021. `https://docs.gro.xyz/gro-docs/archives/introduction-to-defi/insurance`

[59] Grzegorz Halaj. 2018. Agent-based model of system-wide implications of funding risk. (2018).

[60] Hans Haller. 1994. Collusion properties of values. *International Journal of Game Theory* 23, 3 (1994), 261–281.

[61] Robin Hanson et al. 2003. Shall we vote on values, but bet on beliefs. *Journal of Political Philosophy* (2003).

[62] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[63] Stephen J Hardiman, Nicolas Bercot, and Jean-Philippe Bouchaud. 2013. Critical reflexivity in financial markets: a Hawkes process analysis. *The European Physical Journal B* 86, 10 (2013), 1–9.

[64] Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. 2022. Risks and Returns of Uniswap V3 Liquidity Providers. *arXiv preprint arXiv:2205.08904* (2022).

[65] Peter Hewitt. 2010. The future of futarchy. `https://torontopm.wordpress.com/2010/01/04/the-future-of-futarchy/`

[66] Joanne M. Hill. 2012. The Foundation for Enduring Derivative Markets: Liquidity, Transparency, and Capacity. *The Journal of Derivatives* 20, 1 (2012), 14–18. `https://doi.org/10.3905/jod.2012.20.1.014` arXiv:https://jod.pm-research.com/content/20/1/14.full.pdf

[67] Max Holloway. 2022. DYDX Trading Rewards Mechanism Review. `https://mirror.xyz/xenophonlabs.eth/I7BeQC37w3RxH3rxQD5nNznmr3-20rDlh5v1naye68w`

[68] Athul Paul Jacob, David J Wu, Gabriele Farina, Adam Lerer, Hengyuan Hu, Anton Bakhtin, Jacob Andreas, and Noam Brown. 2022. Modeling strong and human-like gameplay with KL-regularized search. In *International Conference on Machine Learning*. PMLR, 9695–9728.

[69] Ifeanyi Jesse. 2022. Love-001 looks uncertain after Makerdao rejection. `https://coinscreed.com/love-001-looks-uncertain-after-makerdao-rejection.html`

[70] John H Kagel and Alvin E Roth. 2020. *The handbook of experimental economics, volume 2*. Princeton university press.

[71] Ehud Kalai and Dov Samet. 1987. On weighted Shapley values. *International journal of game theory* 16, 3 (1987), 205–222.

[72] Hsien-Tang Kao, Tarun Chitra, Rei Chiang, and John Morrow. 2020. An analysis of the market risk to participants in the compound protocol. In *Third International Symposium on Foundations and Applications of Blockchains*.

[73] Dongcheol Kim and Jack Clark Francis. 2013. *Modern portfolio theory: Foundations, analysis, and new developments*. John Wiley & Sons.

[74] Jyrki Kivinen and Manfred K Warmuth. 1999. Averaging expert predictions. In *European Conference on Computational Learning Theory*. Springer, 153–167.

[75] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. 2022. *Towards a Theory of Maximal Extractable Value I: Constant Function Market Makers*. Technical Report. arXiv. org.

[76] Guillaume Lambert and Jesper Kristensen. 2022. Panoptic: a perpetual, oracle-free options protocol. *arXiv preprint arXiv:2204.14232* (2022).

[77] Lyllah Ledesma. 2021. Bitcoin Flash crashed to $5k on Pyth Network's data feed. `https://www.coindesk.com/markets/2021/09/22/bitcoin-flash-crashes-to-5k-on-pyth-networks-data-feed/`

[78] Ian Lee. 2022. Ribbon finance: Decentralized crypto structured products. `https://blog.bybit.com/en-US/post/ribbon-finance-decentralized-crypto-structured-products-blt6c4b4ce0436c1949/`

[79] Hayne E Leland. 1980. Who should buy portfolio insurance? *The Journal of Finance* 35, 2 (1980), 581–594.

[80] Hayne E Leland and Mark Rubinstein. 1988. The evolution of portfolio insurance.

[81] Yun Li. 2021. Options trading activity hits record powered by retail investors, but most are playing a losing game. `https://www.cnbc.com/2021/12/22/options-trading-activity-hits-record-powered-by-retail-investors.html`

[82] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 2031–2063.

[83] Anqi Liu, Mark Paddrik, Steve Y Yang, and Xingjia Zhang. 2017. Interbank contagion: An agent-based model approach to endogenously formed networks. *Journal of Banking & Finance* (2017).

[84] Miguel Sousa Lobo, Maryam Fazel, and Stephen Boyd. 2007. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research* 152, 1 (2007), 341–365.

[85] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 56–67.

[86] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`

[87] Amal M. 2022. Treasury Management in DeFi: Magic of fixed income. `https://medium.com/element-finance/treasury-management-in-defi-magic-of-fixed-income-85c5d2188953`

[88] Fernando Martinelli and Nikolai Mushegian. 2019. Balancer: A Non-Custodial Portfolio Manager, Liquidity Provider, and Price Sensor. (2019).

[89] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[90] Rei Melbardis. 2021. Investing the capital pool in steth. `https://forum.nexusmutual.io/t/investing-the-capital-pool-in-steth/510`

[91] R.C. Merton. 1969. Lifetime Portfolio Selection under Uncertainty: the Continuous-Time Case. *The Review of Economics and Statistics* 51, 3 (Aug. 1969), 247–257. `https://doi.org/10.2307/1926560`

[92] R.C. Merton. 1971. Optimum consumption and portfolio rules in a continuous-time model. *Journal of Economic Theory* 3, 4 (1971), 373–413. `https://doi.org/10.1016/0022-0531(71)90038-X`

[93] Vishal Misra, Stratis Ioannidis, Augustin Chaintreau, and Laurent Massoulié. 2010. Incentivizing peer-assisted services: A fluid Shapley value approach. *ACM SIGMETRICS Performance Evaluation Review* 38, 1 (2010), 215–226.

[94] Nicholas Moehle, Stephen Boyd, and Andrew Ang. 2021. Portfolio performance attribution via Shapley value. *arXiv preprint arXiv:2102.05799* (2021).

[95] Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Perolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, et al. 2019. A generalized training approach for multiagent learning. *arXiv preprint arXiv:1909.12823* (2019).

[96] Bridge Mutual. 2021. Bridge mutual whitepaper V1 released. https://bridgemutual.medium.com/bridge-mutual-whitepaper-v1-released-6004fe158c5a

[97] Vardan Nadkarni. 2022. Polygon Hermez-scaling Ethereum using ZK ROLLUPS. https://medium.com/@vardan.nadkarni/polygon-hermez-scaling-ethereum-using-zk-rollups-eabb1c3603a5

[98] Brian Newar. 2022. Insurace says it will pay millions to claimants after Terra's collapse. https://cointelegraph.com/news/insurace-says-it-will-pay-millions-to-claimants-after-terra-s-collapse

[99] Eric Neyman and Tim Roughgarden. 2021. From Proper Scoring Rules to Max-Min Optimal Forecast Aggregation. *arXiv preprint arXiv:2102.07081* (2021).

[100] Eric Neyman and Tim Roughgarden. 2022. Are You Smarter Than a Random Expert? The Robust Aggregation of Substitutable Signals. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC '22).* Association for Computing Machinery, New York, NY, USA, 990–1012. https://doi.org/10.1145/3490486.3538243

[101] Eric Neyman and Tim Roughgarden. 2022. No-Regret Learning with Unbounded Losses: The Case of Logarithmic Pooling. *arXiv preprint arXiv:2202.11219* (2022).

[102] Emily Nicolle. [n.d.]. DeFi Is Put to the Test and Found Wanting Again. https://www.bloomberg.com/news/newsletters/2022-06-21/defi-is-put-to-the-test-at-solend-and-found-wanting-again

[103] Bernhard Nietert. 2003. Portfolio insurance and model uncertainty. *OR Spectrum* 25, 3 (2003), 295–316.

[104] Peter Nystrup, Stephen Boyd, Erik Lindström, and Henrik Madsen. 2019. Multi-period portfolio selection with drawdown control. *Annals of Operations Research* 282, 1-2 (2019), 245–271.

[105] Geaórid O'Brien, Abbas El Gamal, and Ram Rajagopal. 2015. Shapley value estimation for compensation of participants in demand response programs. *IEEE Transactions on Smart Grid* 6, 6 (2015), 2837–2844.

[106] Yearn State of the Vaults. 2021. The vaults at Yearn. https://medium.com/yearn-state-of-the-vaults/the-vaults-at-yearn-9237905ffed3

[107] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 8024–8035.

[108] David M Pennock and Rahul Sami. 2007. Computational aspects of prediction markets. *Algorithmic game theory* (2007), 651–674.

[109] Jack Peterson and Joseph Krug. 2015. Augur: a decentralized, open-source platform for prediction markets. *arXiv preprint arXiv:1501.01042* 507 (2015).

[110] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. 2019. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* (2019).

[111] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. 2022. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing* 70 (2022), 1142–1154.

[112] Wade Prospere. 2022. Understanding perpetual vaults in DEFI: Structured products with underlying options strategies. https://medium.com/opyn/understanding-perpetual-vaults-in-defi-structured-products-with-underlying-options-

[113] Mellow Protocol. 2021. Uniswap v3: Liquidity providing 101. https://mellowprotocol.medium.com/uniswap-v3-liquidity-providing-101-f1db3822f16d

[114] psykeeper. [n.d.]. Introduction to Saffron Finance. https://docs.saffron.finance/

[115] Anita Ramaswamy. 2022. CEGA raises $4.3m seed round at $60m valuation to build exotic defi derivatives. https://techcrunch.com/2022/03/08/cega-raises-4-3m-seed-round-at-60m-valuation-to-build-exotic-defi-derivatives/

[116] Ronald Ratcliffe, Paolo Miranda, and Andrew Ang. 2017. Capacity of smart beta strategies from a transaction cost perspective. *The Journal of Index Investing* 8, 3 (2017), 39–50.

[117] Max Resnick, Raouf Ben-Har, Drew Patel, and Atul Bipin. 2022. Risk Harbor V2. https://github.com/Risk-Harbor/RiskHarbor-Whitepaper

[118] Dan Robinson, Hayden Adams, and Dave White. 2021. TWAMM. https://www.paradigm.xyz/2021/07/twamm

[119] Aviv Rosenberg and Yishay Mansour. 2019. Online convex optimization in adversarial markov decision processes. In *International Conference on Machine Learning*. PMLR, 5478–5486.

[120] Alvin E Roth. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

[121] Scroll. 2022. Scroll: A layer-2 ecosystem based on ZK-Rollup. https://scroll-zkp.medium.com/scroll-a-layer-2-ecosystem-based-on-zk-rollup-186ff0d764c

[122] Steven Seb. 2022. Lite update: Securing the lite vaults and Steth Peg. https://medium.com/instadapp/instadapp-lite-update-securing-the-lite-vaults-and-steth-peg-3ac35f088a0b

[123] Aleksandrs Slivkins et al. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12, 1-2 (2019), 1–286.

[124] StarkWare. 2022. Starks, StarkEx, and StarkNet. https://medium.com/starkware/starks-starkex-and-starknet-9a426680745a

[125] Estelle Sterrett, Waylon Jepsen, and Evan Kim. 2022. Replicating Portfolios: Constructing Permissionless Derivatives. *arXiv preprint arXiv:2205.09890* (2022).

[126] Yi Sun and Mukund Sundararajan. 2011. Axiomatic attribution for multilinear functions. In *Proceedings of the 12th ACM conference on Electronic commerce*. 177–178.

[127] Mukund Sundararajan and Amir Najmi. 2020. The many Shapley values for model explanation. In *International conference on machine learning*. PMLR, 9269–9278.

[128] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*. PMLR, 3319–3328.

[129] DeepDAO Team. 2022. DeepDAO: DAO Treasuries. https://deepdao.io/organizations

[130] Uniswap Team. 2022. The dominance of Uniswap V3 liquidity. https://uniswap.org/blog/uniswap-v3-dominance

[131] Wealthfront Team. 2022. Investing guide. https://www.wealthfront.com/investing-guide

[132] Tokeninsight. 2022. DeFi yield aggregator yearn.finance launched NFT Treasury Management Tool NFTreasury. https://tokeninsight.com/en/news/defi-yield-aggregator-yearn.finance-launched-nft-treasury-management-tool-nftreasury

[133] N. Torre. 1997. Barra market impact model handbook. (1997).

[134] Mustafa U Torun, Ali N Akansu, and Marco Avellaneda. 2011. Portfolio risk in multiple frequencies. *IEEE Signal Processing Magazine* 28, 5 (2011), 61–71.

[135] Ruby C Weng and Chih-Jen Lin. 2011. A Bayesian approximation method for online ranking. *Journal of Machine Learning Research* 12, 1 (2011).

[136] Whiterabbit. 2020. Black Thursday for Makerdao: $8.32 million was liquidated for 0 Dai. https://medium.com/@whiterabbit_hq/black-thursday-for-makerdao-8-32-million-was-liquidated-for-0-dai-36b83cac56b6

[137] Liam 'Akiba' Wright. 2022. Insurance protocols are paying out millions to UST holders. `https://cryptoslate.com/insurance-protocols-are-paying-out-millions-to-ust-holders/`

[138] Steve Yang, Mark Paddrik, Roy Hayes, Andrew Todd, Andrei Kirilenko, Peter Beling, and William Scherer. 2012. Behavior based learning in identifying high frequency trading strategies. (2012), 1–8.

[139] Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou Ammar. 2019. α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation. *arXiv preprint arXiv:1909.11628* (2019).

[140] Yi Zhang, Xiaohong Chen, and Daejun Park. 2018. Formal Specification of Constant Product (Xy=k) Market Maker Model and Implementation. (2018).

[141] Chunsheng Zhou. 1997. A jump-diffusion approach to modeling credit risk and valuing defaultable securities. *Available at SSRN 39800* (1997).

A CFMM Cost of Manipulation

In this section, we demonstrate that general CFMM portfolio manipulation costs are similar to the cost of Uniswap manipulation. Recall that the portfolio value function [11] determines the value that an LP realizes and the cost of manipulation can be measured by bounds on the portfolio value function change [9]. The portfolio value function is therefore equal to

$$V(p) = \sup_{\lambda \geq 0} \left\{ \lambda\varphi(R) - \lambda(-\varphi)^* \left(-\frac{p}{\lambda} \right) \right\}. \tag{10}$$

where the function $(-\varphi)^*(p)$ is the following:

$$(-\varphi)^*(p) = \sup_{R \in T} \{p^\top R + \varphi(R)\} \quad \text{where} \quad T = \{R | \nabla\varphi(R) = k\}$$

Suppose we send the reserves $R \in \mathbb{R}^n_+$ to $R + R'$, $R' \in \mathbb{R}^n_+$. That is, define the perturbed function $\widetilde{(-\varphi)}^*(p)$

$$\widetilde{(-\varphi)}^*(p) = \sup_{R \in T} \{p^\top (R + R') + \varphi(R + R')\}$$

By the μ-smoothness of the function φ (that is, $\varphi(R + R') - \varphi(R) \geq \mu\|R'\|^2$ for $\mu > 0$), we have the following inequality on $\widetilde{(-\varphi)}^*(p)$:

$$\begin{aligned}
\widetilde{(-\varphi)}^*(p) &= \sup_{R \in T} \{p^\top (R + R') + \varphi(R + R')\} & (11) \\
&\geq \sup_{R \in T} \{p^\top R + \varphi(R) + \mu\|R'\|^2 + p^\top R'\} & (12) \\
&= (-\varphi^*)(p) + \mu\|R'\|^2 + p^\top R' & (13)
\end{aligned}$$

Now, we define the perturbed portfolio value function:

$$\begin{aligned}\tilde{V}(p) &= \sup_{\lambda \geq 0}\{\lambda\varphi(R+R') - \lambda\widetilde{(-\varphi)}^*\left(-\frac{p}{\lambda}\right)\} \\ &= \sup_{\lambda \geq 0}\{\lambda(\varphi(R+R') - \varphi(R) + \varphi(R)) - \lambda\widetilde{(-\varphi)}^*\left(\frac{-p}{\lambda}\right)\}\end{aligned}$$

Applying the smoothness inequality and (11), we have:

$$\begin{aligned}\tilde{V}(p) &= \sup_{\lambda \geq 0}\{\lambda(\varphi(R+R') - \varphi(R) + \varphi(R)) - \lambda\widetilde{(-\varphi)}^*\left(\frac{-p}{\lambda}\right)\} \\ &\leq \sup_{\lambda \geq 0}\{\lambda(\mu\|R'\|^2 + \varphi(R)) - \lambda(-\varphi)^*\left(\frac{-p}{\lambda}\right) - \lambda\mu\|R'\|^2 - p^\top R'\} \\ &= \sup_{\lambda \geq 0}\{\lambda\varphi(R) + \lambda\mu\|R'\|^2 - \lambda\mu\|R'\|^2 - \lambda(-\varphi)^*\left(\frac{-p}{\lambda}\right) - p^\top R\} \\ &\leq \sup_{\lambda \geq 0}\left\{\lambda\varphi(R) + \lambda(-\varphi)^*\left(\frac{-p}{\lambda}\right)\right\} - p^\top R \\ &= V(p) - p^\top R'\end{aligned}$$

This gives us the final bound:

$$V(p) - \tilde{V}(p) \geq p^\top R'$$

Interpreting this bound, a positive manipulation of the reserves changes the portfolio value *linearly* in general CFMMs.

B Staking Model

In order to calibrate and design the Aera token, one needs to consider constraints on slashing, inflation, and collateral requirements as a function of on-chain data. As Aera can be viewed as an adversarial-resistant version of a follow-the-regularized-leader (FoReL) online optimization algorithm, it is natural to cast performance in terms of quantities such as regret and cumulative regret. This document aims to formalize some of these notions and provide inequalities and/or constraints that need to hold in order for the vault to be profitable. We will try to describe these desiderata in words first before writing out the precise constraints.

Notation. For simplicity, we will start by assuming that there is a fixed number of submitters, n, and a fixed number of parameters, k. We will assume there are a assets in the vault with the constraint that $a \leq k$ (e.g. no fewer parameters than assets). We're going to assume there is a block height $h \in \mathbf{N}$ which serves as an index of on-chain data, whereas a continuous time $t \in \mathbf{R}$ will be used for off-chain parameters (e.g. prices). For simplicity, we will assume that the epoch length E is 1. We'll define the variables of interest below:

- $\Delta^n = \{p \in \mathbf{R}^n : p_i > 0,\ \sum_{i=1}^n p_i = 1\}$
- $\pi(h) \in \Delta^n$: Stake distribution over n submitters, $\pi_i(h)$ is the percentage of stake held by agent i at block height h
- $S(h) \in \mathbf{R}_+$: Supply of the token at block height h
- $\xi(h) \in [0,1]$: Percentage of supply staked
- $w_i(h) \in \mathbf{R}^k$: Parameter vector submitted by the ith staker submitter
- $w^*(h) \in \mathbf{R}^k$: Aggregated parameter at height h
- $p(t) \in \mathbf{R}^a$: Price time series of the a assets helds
- $\omega(t) \in \Delta^n$: Reputation score (for weighted median)
- $L(t) \in \Delta^n$: Loan book delta in numéraire terms
- $\Lambda(w(h-1), p(t), L(t)) \in \mathbf{R}$: Objective function of prior weights, current prices and current loan book ($t \geq h$).

We assume the objective function is denominated in numéraire terms and represents the net exposure of the protocol (e.g. portfolio value of a replicating portfolio). As such, we will take the convention of having a goal of *minimizing* the objective function instead of maximizing it. We also assume there exists a metric $d : \mathbf{R}^k \times \mathbf{R}^k \to \mathbf{R}$ that measures distance between submissions. This metric is used to determine if a submission is too far from the existing threshold (which would require the submitter to post more collateral). Parameters associated to this distance measurement include:

- $d^* \in \mathbf{R}_+$: Distance threshold
- $\mathcal{N}(h) \subset [n]$: Set of submission parameters that exceed the distance threshold
- $c \in \mathbf{R}_+$: Collateral ratio to be posted

The set of submission parameters that exceed the threshold is defined as $\mathcal{N}(h) = \{i : d(w_i(h), w^*(h)) > d^*\}$. If a user $i \in \mathcal{N}(h)$, they are required to post $c(d(w_i(h), w^*(h)) - d^*)$ units of collateral.

Objective Function: Bounds and Constraints. In FoReL, two common measures of optimization performance are regret and cumulative regret. These are 'look back' measures that measure how well an action that is taken at time t compares to the best possible action we could have taken at time t, when measured at some time $t' > t$ in the future. In our notion, the k-regret is defined as

$$R_k(h) = \Lambda(w^*(h), p(h+k), L(h+k)) - \min_w \Lambda(w, p(h+k), L(h+k))$$

and the k-cumulative regret is defined as

$$CR_k(h) = \sum_{h'=h}^{h+k} \Lambda(w^*(h), p(h+k), L(h+k)) - \min_w \sum_{h'=h}^{h+k} \Lambda(w, p(h'), L(h'))$$

We will focus on optimizing 1-regret in this document, as it is the only way that we can attribute parameter changes to stakers. However, it is likely that cumulative regret (or a discounted analogue, such as those used in reinforcement learning) can likely have attribution and we leave this for future work.

Lower bound on capital required from submitters. Since the function Λ represents a net exposure, our goal is to penalize submitters who submit a parameter vector w_i that biases the final result significantly. Define $w^*_{-i}(h)$ to be the aggregated parameter vector computed from $(w_1(h), \dots, w_{i-1}(h), w_{i+1}(h), \dots, w_n(h)) \in (\mathbf{R}^k)^{n-1}$. Our goal is to ensure that the minimum collateral that a user has to place if they provide a parameter $w_i(h)$ is

$$c \geq \max_{\ell \in [n]} \mathbf{E}[| \min(\Lambda(w^*(h-1), p(h), L(h)) - \Lambda(w^*_{-\ell}(h-1), p(h), L(h)), 0)|]$$

In words, this says that the amount of collateral required from user ℓ has to be greater than the incremental expected cost that the vault has to pay to cover for user ℓ's parameter submission. For instance, if a user's submission w_ℓ causes the objective function to increase by \$100, then the amount of collateral they must place must be at least \$100. Suppose that Λ is L-smooth in the weight component, e.g.

$$|\Lambda(w, p(t), L(t)) - \Lambda(w', p(t), L(t))| \geq Ld(w, w')$$

then we have

$$\boxed{c \geq L \max_{\ell \in [n]} d(w^*(h), w^*_{-\ell}(h)}$$

For a linear objective function, such as the one used in Pyth [13], the collateral requirement can be computed exactly. However, for us, we will need to estimate L numerically and use that to help set collateral requirements.

Upper bounds on collateral required by submitters. On the other hand, we also have a goal of ensuring that the user does not have to place more collateral than their expected rate of return from submitting to the vault. This effectively is a capital efficiency constraint that says that c must be upper bounded by some notion of rate of return of the vault. One somewhat trivial upper bound states that the maximum collateral that a user provides should be no less than the best possible improvement that a user could have made

$$c \leq \Lambda(w^*(h-1), p(h), L(h)) - \min_w \Lambda(w, p(h), L(h)) = R_k(h)$$

This bound effectively says a user should never have to pay more than the worst case regret. Of course, this bound is often quite pessimistic (e.g. consider the online learning of an equilibrium in a non-no-regret learning game) and it would serve us well to find simpler proxies. Suppose that we have a reward vector $R(\omega(h), \Lambda(w(h-1), p(t), L(t)), \pi(h)) \in \mathbf{R}^n$ that represents the quantity of rewards received by agent i for their update. The network supply updates as

$$S(h) = S(h-1) + \sum_{i=1}^{n} R(\omega(h), \Lambda(w(h-1), p(t), L(t)), \pi(h))_i$$

with the constraint that

$$\sum_{i=1}^{n} R(\omega(h), \Lambda(w(h-1), p(t), L(t)), \pi(h))_i \leq kS(h-1)$$

(e.g. only at most $k\%$ of the supply can be burned in a single time step). We define the expected future ROI of validator i at height h as

$$\mathsf{ROI}(h,T)_i = 1 + \mathop{\mathbf{E}}_{p(t),L(t)} \left[\sum_{t=0}^{T-1} \gamma^i \frac{R(\omega(h+t), \Lambda(w(h+t-1), p(h+t), L(h+t)), \pi(h+t))_i}{\xi(h+t)S(h+t)\pi(h+t)_i} \right]$$

for a discount factor $\gamma \in (0,1]$. We can require that the collateral placed is always less than the expected return

$$\boxed{c < \min_{i \in [n]} \mathsf{ROI}(h,T)_i \xi(h) S(h) \pi(h)_i}$$

C Security Properties of Aera Vaults

From an economic security standpoint, parameter submitters are required to stake AERA to increase the opportunity cost of parameter manipulation. This is necessary because the parameter submitter and the arbitrageur can collude to drain the vault, much like front-running in ETF arbitrage. The inflation emissions of the system are designed to be zero-sum between arbitrageurs and parameter submitters, so that if a parameter submitter is arbitraging their own choice of parameter, they lose out on inflation. More precisely, suppose that parameter θ_h is chosen at epoch h (e.g. block hE) and that at the end of the epoch (block height $(h+1)E$), the parameter submitter receives $i(E)$ units of freshly minted AERA. If an arbitrageur is able to make trades worth τ Aera on blocks between hE and $(h+1)E$, then parameter submitters only receive $i(E) - \tau$ units of AERA at the end of the epoch. Note that if $\tau > i(E)$, then the parameter submitters are slashed and will collectively lose $\tau - i(E)$ units of AERA.

However, this construction is only zero-sum for parameter submitters and arbitrageurs if they use AERA as their numéraire. Suppose the parameter submitter and the arbitrageur are colluding and wish to acquire vault assets for cheaper than fair value by manipulating the parameters θ_h, r_h. If the parameter submitter and arbitrageur are able to find parameters $\theta_h^{adv}, r_h^{adv}$ that are adversarial in that they have the vault rebalance too frequently. This can allow the arbitrageur to trade against the vault frequently and potentially at a substantial discount to market price. Let $\alpha(E)$ be the maximum possible arbitrage profit that can be extracted given the USD prices of the assets $p_{T_1}(t), \ldots, p_{T_n}(t)$ and the adversarial parameters, e.g.

$$\alpha(E) = \sup_{\theta_h^{adv}, r_h^{adv}} \sum_{i=hE}^{(h+1)E} \rho(\theta_h^{adv}, r_h^{adv}, p_{T_1}(i), \ldots, p_{T_{n(i)}})$$

where ρ is a rational arbitrageur's USD PNL for trading at block height i given the parameters and USD prices. If $p_{\text{AERA}}(h)$ be the price of AERA in USD at block height h, then the system is insecure to collusion between the arbitrageur and parameter submitter if

$$\alpha(E) > i(E)p_{\text{AERA}}(h)$$

In practice, however, neither the arbitrageur nor the parameter submitter can precisely estimate $\alpha(E)$. Instead, they can only estimate an expected arbitrage profit in epoch E, $\hat{\alpha}(E)$:

$$\hat{\alpha}(E) = \sup_{\theta_h^{adv}, r_h^{adv}} \sum_{j=0}^{E} \sum_{i=hE+j}^{(h+1)E} \gamma^i \, \mathbf{E}\left[\rho(\theta_h^{adv}, r_h^{adv}, p_{T_1}(i), \ldots, p_{T_n}(i)) | p_{T_k}(l) \forall k \in [n], \forall l < i\right]$$

where $\gamma \in (0, 1)$ is a discount factor. Note that this is quite similar in spirit to a value function from reinforcement learning. Our goal is to show that the protocol is safe in a Value-at-Risk (VaR) sense: Given a vault with a USD portfolio value PV, the probability of arbitrage profits being greater than $i(E)p_{\text{AERA}}(h)$ is less than α. Formally, the protocol is *(α, ϵ)-VaR-safe* if

$$\mathbf{Prob}\left[\hat{\alpha}(E) - i(E)p_{MAX} > (1 - \epsilon)PV\right] < \alpha$$

As an example a $(0.0001, 0.05)$-VaR-Safe portfolio has less than a 0.01% chance of losing more than 95% of the vault portfolio value to arbitrageurs.

To achieve an (α, ϵ)-VaR-safe protocol, we will make it expensive for a parameter submitter to submit a malicious parameter. We use a control mechanism inspired by adversarial online convex optimization literature. Suppose that we have a two convex loss functions, $\ell_\theta(\theta_{old}, \theta_{new})$ and $\ell_r(r_{old}, r_{new})$. If a parameter submitter submits a parameter θ_{h+1}^{adv} at epoch $h + 1$ such that either $\ell_\theta(\theta_h, \theta_{h+1}^{adv}) > \zeta_\theta$ then the submitter has to add collateral to the vault. In particular, the submitter has to add

$$\frac{\ell_\theta(\theta_h, \theta_{h+1}^{adv}) - \zeta_\theta}{p_{\text{AERA}}(t)}$$

of stablecoins to submit θ_{h+1}^{adv} to the protocol. Similarly, there is $\zeta_r > 0$ such that the same is true for a submitter to submit r_{h+1}^{adv} to the protocol. The extra collateral posted is meant to disincentivize collusion between the arbitrageur and the submitter for one epoch. If $\alpha(E) < i(E)p_{\text{AERA}}(h)$ (as estimated by the protocol), then the submitters collateral is returned to them with a small percentage of bonus AERA. The bonus is added so that a submitter is incentivized to submit a large parameter change if needed (e.g. non-adversarial) and to incentivize the opportunity cost of capital in this scenario. Our claim is that with judicious choices of $\ell_\theta, \ell_r, \zeta_\theta, \zeta_r$ as a function of α and ϵ, the protocol can achieve (α, ϵ)-VaR-safety.

D Fee Attribution and Slashing

In this section, we will provide a more formal description of the Shapley value calculation used and notes on the precise fee distribution mechanism. We also describe the baseline comparison problem [126, 127]. The majority of our notation will be akin to [127], which we refer the reader to for further technical details.

Definition of Shapley Values for an objective function. Given a set N, the binary Shapley value Λ_i associated to a set function $v : 2^N \to \mathbf{R}$ and an element $i \in N$ is defined as

$$\Lambda_i = \sum_{S \subseteq N - \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S))$$

For a continuous function f, such as a DAO's objective function f_{DAO}, we can construct the baseline set function v_B as

$$v_B(S) = f(x_S; x'_{N-S})$$

where $x' \in \mathbf{dom}\, f$ is the *baseline* and the vector $x = (x_S; x'_{N-S}) \in \mathbf{R}^n$ is the concatenation of the elements of x, x' but only supported on S and N, respectively. Associated to this is the baseline Shapley value, $\Lambda_i^B(S)$,

$$\Lambda_i^B = \sum_{S \subseteq N - \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v_B(S \cup \{i\}) - v_B(S))$$

Finally, note that one can expect the baseline Shapley value to the *random baseline Shapley value*, $\Lambda_i^{B,D}(S)$ which is associated to the set function

$$v_{B,D}(S) = \mathop{\mathbf{E}}_{x' \sim D}[f(x_S; x'_{N-S})]$$

where D is a distribution over admissible baselines.

Shapley Values in Aera. For Aera, the set N represents the set of submitters, S represents a subset of submitters, and x_S represents the aggregated portfolios of a subset of submitters. We can view x' as a baseline portfolio that we're grading submitters relative to and view the vector $(x_S; x'_{N-S})$ as stating that submitters $S \subset N$ had their parameters aggregated whereas the submitters $N - S$ did not submit parameters. Using the definition of the binary Shapley value, one can view $\Lambda_i^B(S)$ as looking at the expected marginal impact of i's submitted portfolio on all other subsets of submitters values. There are a number of applications where it makes sense to directly use the baseline Shapley value to grade users, if we have a benchmark. For instance, if an Aera vault's objective function is to optimize stablecoin-denominated yield, then the benchmark x' can simply be a weighted average of other yield sources within DeFi (*e.g.* Yearn Finance). On the other hand, a benchmark may also be stochastic to adjust for the fact that we do not have perfect counterfactual information about how the vault would have performed under different circumstances. In the lending coverage usage of Aera, it might make sense to average over various volatilities to construct a benchmark to compare to. The precise usage of Shapley values in each Aera vault will depend on the particular objective function and its sensitivities.

Using Shapley Values to choose fees As described in the introduction, we can use the Shapley values to redistribute fees according to submission quality and stake. The first necessary condition for any fee distribution mechanism is that the total weighted sum of rewards should sum to zero (equivalent to a balanced budget condition). Let $\Lambda'_i = \Lambda_i - (\Lambda/n)$ where $\Lambda = \sum_i \Lambda_i$, where Λ_i is the Shapley value associated to agent i. Then we can use the following fee distribution mechanism (which is a generalization of the formula in the introduction):

$$\gamma(h)_i = s_i(h)\frac{\Lambda'_i}{\max_i |\min(\Lambda'_i, 0)|} + \gamma(h)\frac{\max(\Lambda_i, 0)}{\sum_i \max(\Lambda_i, 0)}$$

where $s_i(h)$ is the stake of the i agent, $\gamma(h)$ are the fees to be distributed at height h, and Λ_i are the baseline Shapley values. However, we note that this model completely slashes the stake of the worst case submitter, which can lead to other undesirable effects [29, 33]. If we do not completely slash the agents submitting in an epoch, we will have to send the remaining fees elsewhere to ensure the zero-sum condition holds. One possibility is that this execess penalty from slashing goes directly towards the DAO treasury with disbursements of lost stakes reaching the other submitters only over time via emissions. Then we would utilize the fee distribution

$$\gamma(h)_i = \frac{\gamma(h)}{\sum_i |\Lambda_i|}\max\left(\Lambda_i \cdot \frac{\sum_i |\Lambda_i|}{\sum_i \max(\Lambda_i, 0)}, 0\right)$$

Note that this decomposition is not unique and is one of many possible ways to weight contributions taking the magnitude and distribution of the negative Shapley values into account.

In general, fee distribution shouldn't necessarily be fixed every block or epoch as some updates are more valuable than others or around times that are more valuable (and/or have higher returns to depositing DAOs). As such, there is some sense that the rewards should be tied to some measure of the total value Λ. Another related concept is that the grading of the submissions and how rewarding the best submitter out of a bunch of bad submissions isn't necessarily desirable whereas penalizing the worst submitter out of a bunch of great submissions may be too harsh. This can be done by introducing a dynamic fee penalty, where the penalty for a bad submission increases over time (with the excess revenue going to the DAO).

Another possibility is just using the Shapley values directly for slashing without modification so defining $g(v)$ as some stretched and scaled sigmoidal function

$$\gamma(h)_i = \gamma(h)\frac{\Lambda_i}{\sum_i |\Lambda_i|} g(v)$$

where positive values get rewarded, negative values get slashed and the remainder of $f(h)$ (max distribution is 1) and the slashed stakes go back into the treasury. So in that sense $f(h)$ is more of an upper bound for block reward rather than an expected reward each block and $g(v)$ at least ties the total disbursement to the value v of the aggregate. There is some question of whether the bounding works out so the negative values are always covered by the stake but any shortfall could just be deducted out of the rewards pro-rata which is a corner case that needs to be tested carefully prior to deployment. We note that sigmoidal modification to Shapley values has been studied in a number of contexts [93, 105] and serves as a reasonable heuristic.

Finally, a last formulation for fee sharing involves utilizing $h(L)$ in place of $g(v)$ where $L = f_{\text{DAO}}(\pi_{\text{OPT}}) - f_{\text{DAO}}(\tilde{\pi})$ where h is monotonically decreasing with $h(0) \to 1$ and $h(\infty) \to 0$ and a sigmoidal transition. This corresponds to evaluating a sigmoid at $v \approx f_{\text{DAO}}(\tilde{\pi}) - f_{\text{DAO}}(\tilde{\pi}_{t-1})$, which effectively creates a baseline for evaluating the value of the portfolio update versus doing nothing (*e.g.* using the previous epochs update). In practice, the Aera team is evaluating all of these options numerically to arrive at a stable, efficient fee distribution rule. We note that all fee updates are required to be compatible with the slashing and collateral conditions of Appendix B.

E Simulation Technicals

E.1 Loan Book Representation

A basic methodology for computing the loan book portfolio value is adapted from [32]. In this toy setup loans are defined by borrowing asset $\mathcal{B}$ collateralized by supplying asset $\mathcal{A}$. Each loan in the book can be loosely decomposed into a quanto digital call option to the borrower that is struck at insolvency price in $\mathcal{B}$ with an all-or-nothing payout in $\mathcal{A}$ along with a quanto barrier down and in call option to the liquidator with barrier in liquidation price in $\mathcal{B}$, a payout in $\mathcal{A}$ and exercisable between the insolvency and the liquidation price in $\mathcal{B}$.

Note that without liquidators or external changes, borrowers have no impetus to exercise since they can just keep the assets and so they only profit relative to their prior state if the options are out of the money (OTM). Thus liquidators are needed to act as the option exercisers where the value of the barrier option comes out of the borrower option. Under no-arbitrage, given a liquidation bonus ϵ and denoting the value of the barrier option as $\mathcal{V}_{liq}$ then $\lim_{\epsilon \to 0} \mathcal{V}_{liq} = 0$ and for exercise range width ξ the payoffs converge in $\lim_{\xi \to \infty} \mathcal{V}_{liq} + \mathcal{V}_{bor} = \mathcal{V}_{call}$ so the collective value of these options is upper bounded by a call. The total portfolio of the book is then approximated by long supplied collateral, short borrowed assets and short calls.

E.2 Loan Book Portfolio Value Computation

We provide an example for a loan book containing two assets ETH and USDC, noting that a large class of more general lending protocols can have their risk profile projected onto such a portfolio. Consider call and put options with strikes corresponding to the price of ETH in USDC whose payout is denominated in USDC. Then ignoring recursive same asset loans the book can be divided into ETH collateralized USDC loans and USDC collateralized ETH loans where the former can be represented by calls (which we consider to be long risk) for the protocol and the latter can be modeled by puts (which we consider to be short risk).

For each loan (s, b) where s denotes ETH supplied quantity and b denotes USDC borrowed quantity then the strike of the call is given by $K = \frac{b}{s}$ and denoting the present price of ETH as S then assuming a risk-free rate $r = 0$ the value C of the call for some time to expiration t and volatility σ as given by Black-Scholes is

$$C = \Phi(d_1)S - \Phi(d_2)K$$

where

$$d_1 = \frac{\log \frac{S}{K} + \frac{\sigma^2 t}{2}}{\sigma\sqrt{t}}$$

and

$$d_2 = \frac{\log \frac{S}{K} - \frac{\sigma^2 t}{2}}{\sigma\sqrt{t}}$$

and $\Phi(x)$ is the CDF of the standard Normal $\mathcal{N}(0, 1)$ given by

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-z^2/2} dz$$

The total portfolio value V is then given by

$$V = \sum_{(s,b)} (s(S - C) - b)$$

where we note that $V \leq 0$ since the calls collective intrinsic value is equal to the total collateral net total borrow and the remaining time value is nonnegative. This can be thought of as an approximation for the risk the protocol bears in exchange for earning interest from the spread between borrow and supply rates.

In the analogous case where each loan is given by (s, b) where s denotes USDC supplied quantity and b denotes ETH borrowed quantity the strike of the put is given by $K = \frac{s}{b}$ and the value P of the put is

$$P = \Phi(-d_2)K - \Phi(-d_1)S$$

with a total portfolio value V given by

$$V = \sum_{(s,b)} (s - b(S + P))$$

In the context of the simulation, the options are issued to borrowers and liquidators shrink the overall book balance sheet since each liquidation repays outstanding borrow and takes collateral. The magnitude of the portfolio values thus decreases with each liquidation, which is effectively an early exercise of in the money (ITM) borrower options where the liquidation bonus comes out of borrower call gains and borrower option time value is relinquished.

E.3 Vault Portfolio Construction Process

We present a simple formulation for a vault with two assets ETH and USDC, noting that a larger space of asset possibilities allow for increased efficacy of classical methods such as Markowitz mean-variance optimization over the following heuristic based approach. Denoting the starting vault value in USD as Q, weight bounds for USDC given by w_{floor} and w_{ceil} so $0 < w_{\text{floor}} \leq w_{\text{ceil}} < 1$ and the total long portfolio value as V with a risk buffer scalar $c \geq 1$ then the initial weight of USDC

$$\texttt{target\_share2} = \max\left(\min\left(\frac{-cV}{Q}, w_{\text{ceil}}\right), w_{\text{floor}}\right)$$

with the remaining weight allocated towards ETH.

The motivation behind such a construction is to rotate into a primarily stable asset portfolio when long risk has accumulated and there is a higher likelihood of risk off cascading liquidation spirals and also rotate into a portfolio with more ETH and higher risk return when the positional risk is low. Note that this focuses on the long risk due to the absence of short deltas to hedge with while the short risk is often fully covered by the presence of ETH.

When updating weights there is a tradeoff between rotating too frequently and overpaying rebalance costs and rotating too slowly and being run over by adverse market conditions. This naturally introduces a smoothing time window parameter W denoting the number of previous time steps to average over so at each time step T

$$\texttt{target\_share2} = \max\left(\min\left(\frac{-\sum_{t=T-W+1}^{T} cV_t}{\sum_{t=T-W+1}^{T} Q_t}, w_{\text{ceil}}\right), w_{\text{floor}}\right)$$

Additional considerations are weight change bounds per update per time interval. A bound in absolute terms such as

$$\texttt{target\_share2} = \max\big(\min\big(\tilde{w}_T, w_{T-1} + \gamma\Delta T\big), w_{T-1} - \gamma\Delta T\big)$$

where $\tilde{w}_T$ is the intermediary output from the previous equation, γ is a scalar and ΔT is the time elapsed. Another bound is based on vault spot price change where

$$\frac{\tilde{w}_T(1 - w_{T-1})}{(1 - \tilde{w}_T)(w_{T-1})} \leq k\Delta T$$

and

$$\frac{(1-\tilde{w}_T)(w_{T-1})}{\tilde{w}_T(1-w_{T-1})} \le k\Delta T$$

which simplifies to

$$\texttt{target\_share2} = \max\left(\min\left(\tilde{w}_T, \frac{k\Delta T w_{T-1}}{1+(k\Delta T-1)w_{T-1}}\right), \frac{w_{T-1}}{k\Delta T-(k\Delta T-1)w_{T-1}}\right)$$

for some scalar k. Note that if $k \le \Delta T^{-1}$ then no weight changes are possible.

E.4 Objective Function

A rudimentary high-level functional form for an objective function is

$$\alpha \cdot \text{Yield} + \beta \cdot \text{Risk}$$

for some tuned constants α and β for balancing the two components while retaining interpretability. An example yield metric is the vault's return

$$\text{Yield} = \frac{Q_T}{Q_0} - 1$$

where the compounding aspect of APY isn't factored in due to the extrapolation noise. An example risk metric is a risk coverage shortfall relative to vault value percentage stat. Define a risk coverage shortfall event as anytime $Q_t + V_t < 0$ with the indicator variable $\mathbf{1}_{Q_t+V_t<0}$ so

$$\text{Risk} = \frac{\sum_{t=0}^{T}(Q_t+V_t)\cdot\mathbf{1}_{Q_t+V_t<0}}{\sum_{t=0}^{T} Q_t\cdot\mathbf{1}_{Q_t+V_t<0}}$$

In practice the indicator function can be replaced with a smooth sigmoidal proxy such as softmax to prevent numerical artifacts due to the discontinuity. Furthermore, note that the shortfall statistic has a stronger dependence on the initial condition Q_0, whereas the return series does not.