Aquo: DeFi Optimization

Oakley, Trevor Lee

April 29, 2024

Abstract

We propose an optimization architecture for DeFi Protocols to reduce impermanent loss, slippage in trades, and to mitigate risks in liquidity deployment. Our architecture is based on reinforcement learning (RL) systems and DeFi Composition (termed ComFi in this paper).

If successfully implemented, these proposed systems would also allow the building of structured financial products including derivatives, and to combine protocols to deliver a complete suite of financial products in a single transaction to a user.

1 Introduction

Since the 2008 financial crisis [17], a number of financial system innovations happened, these included the introduction of blockchains, smart contracts, Decentralized Finance (DeFi), and DeFi Composition.

DeFi emerged in 2020, and the DeFi protocols have a lot of room for growth. We propose solving the following problems (which are explained more fully in this document) related to DeFi Compositions and Derivatives, which we call ComFi:

- Optimize liquidity aggregation in ComFi markets.
- Reduce risks in settlements in ComFi markets.
- Determine the benefits and suitable of a set of new standards for smart contract interfaces for ComFi.

I. Finance & Financial Systems Background

1.1 Finance and Financial Systems

The subject matter of finance, its history, components, definitions, theories, and scope are long and complex. The relevance to this document is to show the evolution of financial systems which were once recorded on clay tablets (3000 BC in Uruk), [48], to modern global systems processing trillions of dollars a day via a complex arrangement of systems including central banks, commercial banks, stock exchanges, clearing houses, and more.

Financial systems process loans, derivatives, payments, settlements, exchanges, investments, clearing, and money creation (by central banks). These systems work via a larger infrastructure, called the Financial Market Infrastructure (FMI), and a critical part of FMIs is the derivative [24].

1.1.1 Derivatives

One of the focal points to this document is the derivative in emerging financial systems. The modern derivative is pivotal to financial markets. Its history dates back to the reign of the King of Bablyon (Hammurabi) in 1750 BC and his widely cited Code of Hammurabi [42]. Derivatives were widely used during the Roman period, through the Middle Ages, and into the modern period; their growth increased dramatically in the 1980s.

To define the concept of a derivative, we can define an agreed price in the future. Taking the most simple example of a farmer producing some grain. The farmer agrees the price for the grain before the grain is harvested. This guarantees the price to the farmer some months away giving certainty to the farmer. This is called a forward derivative. What makes these financial contracts interesting, is that they can then be traded. Hence when a contract exists to buy grain at 100 dollars per bushel in June, that contract itself can be traded.

If the contract is to buy grain at 100 dollars per bushel then an option to buy the contract can be obtained to buy the contract for 5 dollars. Then if at the time of maturity (e.g. June in this example), the market price is 120 dollars, the option holder can exercise the option and gain 15 dollars. Then of course, there can derivatives on the derivatives. Hence we can then say there is a option to buy the original option contract for 2 dollars. This reduces the loss to 3 dollars to the original option holder, if the second option is not exercised (i.e. their contract rights are exercised).

The significance of derivatives was shown with the Nobel Prize for the Black-Scholes formula [2] in 1997 and today there is a large subject area to model prices in derivatives [43]. Today, derivatives are traded in high volumes, with 6.5 trillion USD a day being recorded in 2019 [20].

1.2 Complex and Structured Financial Products

An important concept in finance and financial systems is the one of a structured financial product. It relates to large capital requirements of corporations and normally will include: Asset Based Securities (ABS), Mortgage Backed Securities (MBS), Collateralized Debt Obligations (CDOs), or Commercial Mortgage Backed Securities (CMBS) [23] [36].

The relevance to this document and emerging financial systems, is that structured financial products are complex by nature. This complexity is one of the most important aspects of a concept to be discussed later called DeFi Composition in DeFi.

We can take one example to show this concept with an ABS structured product. Consider a bank has 100 million USD in car loans, this is then split into tranches: 80M USD, 15M and 5M USD for the senior, mezzanine, and equity tiers. As the loans generate cashflows, the senior tranche investors get paid first, then mezzanine ones, and finally equity ones.

In this way, the bank derisks its own lending and increases liquidity.

1.3 Financial Intermediary

A key concept in financial systems is an intermediary. A financial intermediary is an organization which acts between two parties to facilitate a transaction. An example could be a commercial bank acts between a central bank and a pension fund [9]. Therefore the pension fund can trade with the central bank.

A common misconception is that banks are simply intermediaries, but commercial banks create money (they do not as is commonly believed loan out money which was deposited into the bank) and many economic textbooks are wrong [8].

The relevance of intermediaries and banking will be clear to emerging financial systems later.

1.4 Liquidity

Liquidity is a fundamental term in financial systems but one which is often not understood.

There are different forms of liquidity:

- Banking: A bank's ability to meet its short term obligations
- Asset: Ease and cost of converting an asset into cash [12]
- Market/Trading: Ability to sell an asset without the price being affected (deep liquidity refers to highly liquid markets)

Illiquid markets cause significant asset price moves during trading. Illiquid assets cause investor risk as assets cannot be converted easily.

1.5 Real-World Assets

Like many terms in financial systems, there is no exact or universally agreed definition for the term real-world asset. The parallel and more established term is a real asset which Bodie et al. [12] defines as assets which cause productive capacity in the economy (e.g. land, buildings, knowledge, machines, and employees); this is contrasted to financial assets which are claims on the income generated by real assets (e.g. stocks or bonds) [60].

Real-world assets (RWAs) are typically discussed in terms of digitization [28] and more recently with blockchains. RWA means an asset which exists in the real world (i.e. not digital only) but we can represent it digitally.

In blockchain terms, we typically refer to RWAs in the context of tokenization.

II. Technological Background

1.6 Databases

Data itself was organized and recorded over centuries, with records dating back to 30,000 BC and the ”tally stick.” The tally stick was a major form of record keeping in England which emerged in 1100 AD under King Henry I and then continued until 1826. Notches were made in the wooden tally stick and the size of the notch indicated the size of the denomination. The tally stick was split vertically and given to the two parties as proof of the record [5].

The Dewey Decimal System was conceived in 1873 [53] which was early form of classification which was adopted by 200,000 libraries, and is seen as a forerunner to databases.

Modern databases emerged in the 1960s and 1970s [55] (hierarchical and relational). These databases were centralized. Databases subsequently emerged into many forms including object, object relational, and NOSQL. We can consider three types of databases [58]:

- Centralized (reside on a single storage device)
- Decentralized (can be hierarchical)
- Distributed (data is replicated across devices called nodes)

It is the distributed database which is typically a NOSQL database (key-value store) which is of most interest to this document.

1.7 Distributed Ledger Technology (DLT)

A distributed ledger (DL) is a type of distributed database which assumes the presence of malicious nodes, and data can only be appended or read [57].

A DLT is a data structure which introduces a consensus algorithm so that nodes can agree on a version of truth. Public key cryptography is used to verify the nodes are genuine (nodes have a private and a public key) [21]. DLTs can be linked lists (blockchains), DAGs, or a list of linked lists (sidechains), as follows:

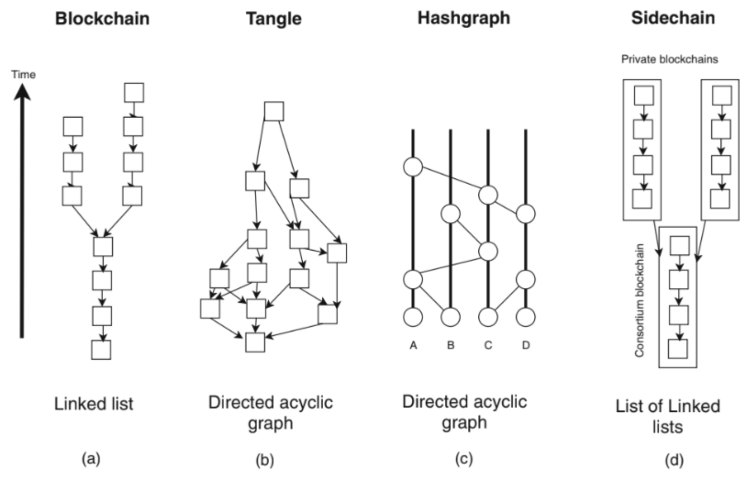


Figure 1: DLTs

Therefore, DLTs allow data to be verified in an untrusted environment via the consensus algorithm and the public key cryptography between nodes.

1.8 Blockchains

Blockchains emerged as a promising form of DLTs in 2009 with the Bitcoin white paper [47]. A blockchain has a series of blocks all interconnected via hashes and each block has transactions in it.

1.8.1 Bitcoin

Bitcoin uses a Unspent Transactions Outputs (UTXO) database to record all the transactions on the DL. It uses a consensus algorithm called Proof-of-Work (PoW). PoW is a cryptographic primitive used for decades to prevent email spam, DoS attacks and sybil attacks [26]. It has a proving and a verification algorithm.

In Bitcoin's PoW system, miners (nodes creating blocks) compete to find a nonce (number used once) that, when combined with the block data and processed through a hash function, results in a hash value that meets the blockchain's network current difficulty level. This process requires significant computational effort, and the first miner to find such a nonce can create a new block and broadcast it to the network. If the block is validated by other nodes, it is added to the blockchain, and the successful miner receives a reward in the form of new bitcoins (the block reward) and the transaction fees from the transactions included in the block.

1.8.2 Ethereum & the EVM

Ethereum [14] extended the concepts of Bitcoin to include an account system on a new blockchain called Ethereum, which also used PoW (initially). Ethereum implemented the Ethereum Virtual Machine (EVM) which runs bytecode (compiled code from a high level language such as solidity). These programs run by the EVM are called smart contracts and they enable programmatic functions to be performed on a blockchain node.

1.8.3 Other Blockchains

Following Ethereum and Bitcoin a large number of other blockchain arose with varying consensus algorithms and claims to scale better, e.g. Cardano, EOS, Near, BNB, Solana, Dogecoin, Avalanche, and many hundreds more.

1.9 Smart Contract

The concept of a smart contract was conceived by Nick Szabo in 1997 [59]. He proposed formulating a contract which could embed property using legal and principles with protocols on a computer network.

With the emergence of blockchains, the smart contract (SC) took the meaning of self-executing code on a blockchain node. Within a SC, there is a state which refers to persistent data and functions stored in the SC. The data in the SC can be property too, such as a cryptoasset [69].

In this sense, the SCs process state and speaking more generally, the blockchain can be considered as a state transition machine. A SC is immutable and once deployed it cannot be changed at that address. A SC forms part of an application referred to as decentralized application (dApp).

1.10 Gas

Gas is a term commonly associated with Ethereum, and it is a fee used to pay the miners [46]. Other blockchains have similar fees but they can just be called a transaction fee.

1.11 Blockchain Address

The address is an identifier which identifies a wallet address and/or a contract address.

1.11.1 Bitcoin

The Bitcoin address used normally as an destination to receive funds (or send funds from). It is worked out as follows [52]:

Step 1. Generation of 256-bits random number - private key.

```
f8f8a2f4 3c8376cc b0871305 060d7b27 b0554d2c c72bccf4 1b270560 8452f315
```

Step 2. Generation of a corresponding public key, with the usage of secp256k1 elliptic curve and ECDSA algorithm.

```
04 6e145cce f1033dea 239875dd 00dfb4fe e6e3348b 84985c92 f1034446
83bae07b 83b5c38e 5e2b0c85 29d7fa3f 64d46daa 1ece2d9a c14cab94 77d042c8
4c32ccd0
```

Step 3. Calculating SHA-256 hash for the result of Step 2.

```
0580c169 481e6d74 856bc146 fbb6ab71 73f7a2fd b6b2a691 4309e04b 7e37d39b
```

Step 4. Calculating RIPEMD-160 hash for the result of Step 3.

```
b49ac57c dd6bb585 56ff60eb 07de6a6b def013d2
```

Step 5. Concatenating prefix 00 to the result of the Step 4.

```
00 b49ac57c dd6bb585 56ff60eb 07de6a6b def013d2
```

Step 6. Calculating SHA-256 hash for the result of Step 5.

```
04695f7e f43f9ce1 aad6c312 3a2f0d8e 8bb3b059 f120b4fa 462438b3 4d3f865b
```

Step 7. Calculating SHA-256 hash for the result of Step 6.

```
38b08e65 175b74a7 7ead88aa 80a1d107 59ab8721 94f741a8 202491ce b81018cd
```

Step 8. Concatenating four the most significant bytes from the result of Step 7 and it's as a suffix to the result of Step 5.

```
00 b49ac57c dd6bb585 56ff60eb 07de6a6b def013d2 38b08e65
```

Step 9. Calculating Base58 value of the result of Step 8.

```
1HTx3EmeoLWX7gcRhKbBWCJ7CsxAoSYRPa
```

1.11.2 Ethereum Blockchains & EVM Chains

There are two main uses of addresses in Ethereum, one for the wallet address (EOA), and one for the contract deployed address. These are 20 byte fields.

Wallet

The public key is then hashed using the Keccak-256 hashing algorithm (part of the SHA-3 family). This hashing process is what creates the Ethereum address.

The last 20 bytes (40 hexadecimal characters) of the hashed public key become the Ethereum address. This truncation ensures the address is of a fixed length and maintains its uniqueness.

Let's assume the hash of the public key is: 0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef

The last 20 bytes of this hash would be: 0x90abcdef1234567890abcdef1234567890abcdef

Prefixing it with '0x' gives us the Ethereum address: 0x90abcdef1234567890abcdef1234567890abcdef

Contract

The contract address is derived from the EOA's address, and the nounce on the account (number of transactions on that account). The resulting string is hashed using keccak256.

$contract\_address = \text{keccak256}\left(\text{RLP\_encode}(sender\_address, sender\_nonce)\right)[12:]$

Here, RLP_encode is the Recursive Length Prefix encoding, and [12:] indicates taking the last 20 bytes (the address length) of the resulting hash.

1.12 Smart Contract Interfaces

A smart contract interface defines the external-facing functionalities and behaviors of a smart contract. It outlines the methods and data structures that other smart contracts or external entities can interact with. Two commons standards in Ethereum are ERC20 and ERC721 which also define interfaces eg ERC20 Interface [49].

1.13 Smart Contract Composition

Smart contract composition (SCC) is different to DeFi Composition (to be discussed later) but the principles of SCC relate to DeFi Composition.

A SC composed transactions is when one SC calls another one. It does this typically via an interface and it uses the address of the deployed contract.

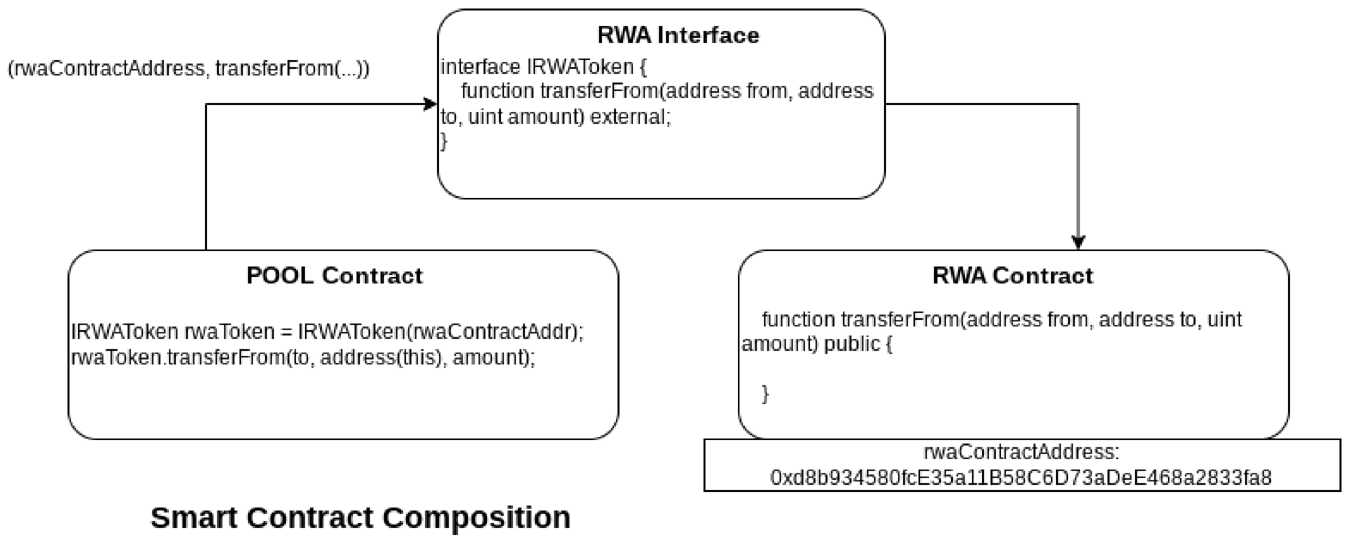
In this manner, one SC can utilize the functionalities of other contract only via a SC call. This expands the capabilities of SCs and reduces costs for developing dApps.

Kairan Sun et al. analysed 350,000 smart contracts deployed on Ethereum during a two year period [56] in the context of a contract calling a subcontract and they found a typical smart contract comprises of 10 subcontracts.

An example follows in solidity showing how the transferFrom can be executed from a Pool Contract on an RWA contract deployed at address rwaContractAddr. A message is sent via the EVM from the calling contract to the called one.

Listing 1: Smart Contract Pool Code for RWA Token Transfer

```
IRWAToken rwaToken = IRWAToken(rwaContractAddr);
rwaToken.transferFrom(to, address(this), amount);
```


RWA Interface
(rwaContractAddress, transferFrom(...))
POOL Contract
RWA Contract
Smart Contract Composition

```
interface IRWAToken {
   function transferFrom(address from, address
to, uint amount) external;
}
```

```
IRWAToken rwaToken = IRWAToken(rwaContractAddr);
rwaToken.transferFrom(to, address(this), amount);
```

```
   function transferFrom(address from, address to, uint
amount) public {

   }
```

rwaContractAddress:
0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8

Figure 2: Smart Contract Composition

1.14 Wallets

Introduction

A wallet generates and holds public and private keys, which are then used to perform operations on a blockchain. There are several types of wallets, e.g. hardware, software, web, cold, hot, and mobile [39].

Wallets normally work with one set of blockchain standards, eg Metamask works with EVM chains, and Bitcoin Core is for Bitcoin.

Hierarchical Deterministic (HD) Wallets

A deterministic wallet usually contains a pair of master keys (msk, mpk) and a seed ch, which is also referred to as the chaincode. For every new transaction, the wallet deterministically derives a fresh session key pair (sk, pk) from the master keys with the help of deterministic key derivation algorithms [19]. A hierarchical deterministic wallet uses a tree to organize keys.

1.15 TradFi, DeFi, and CeFi

The term TradFi was introduced to distinguish older mainstream financial systems from Decentralized Finance (DeFi), DeFi is defined in the DeFi section.

TradFi, short for traditional finance, refers to the financial systems, methods, and procedures all used today in banking, stock exchanges, central banks, insurance companies, payment systems, settlements, clearing houses, depositories, securities, over-the-counter (OTC), debt, derivatives, pensions funds, brokerages, credit unions, regulators, foreign direct investments, cross-border payments, letter of credit, and mortgage companies. TradFi markets are very large (trillions of dollars a day).

CeFi refers to centralized finance which would account for most TradFi processing, but not all. The meanings between TradFi and Cefi are hence slightly different but related.

1.16 Oracles

A blockchain cannot access natively data outside of the blockchain. We use the terms on-chain and off-chain. A solution to bridge data from off-chain to on-chain (e.g. price feeds) is called an oracle.

There are centralized and decentralized oracle networks (DON). One of the main leaders in this space is Chainlink [13].

1.17 Tokenization

Tokenization refers to the process of converting rights to an asset, such as real estate, stocks, or commodities, into digital tokens that can be traded on a blockchain or distributed ledger technology (DLT) platform. These digital tokens represent ownership, equity, or utility over the underlying asset [33].

This is a significant step into evolving blockchain applications because it allows a completely digital representation of property rights on a blockchain which can then be the basis for financial instruments.

III. Emerging Financial Systems

2 DeFi

2.1 Introduction

Decentralized Finance (DeFi) emerged in the summer of 2020 as a challenge to TradFi systems. DeFi uses smart contracts to implement financial functionality.

There are no precise definitions for DeFi which are universally agreed, but the concepts are similar between researchers. We can define DeFi - using [67], [6], and [35] - as follows:

- Non-custodial (user directly manage their wallets)
- Permissionless DLTs
- Auditable (access by anyone to records)
- Composable (based on building blocks)
- Open Source Code
- Has a financial application

2.2 DeFi Infrastructure

There is universal agreement that a DeFi Infrastructure is layered as visualized below [6], and especially of note is composability is listed as a core component in the DeFi Layers model.

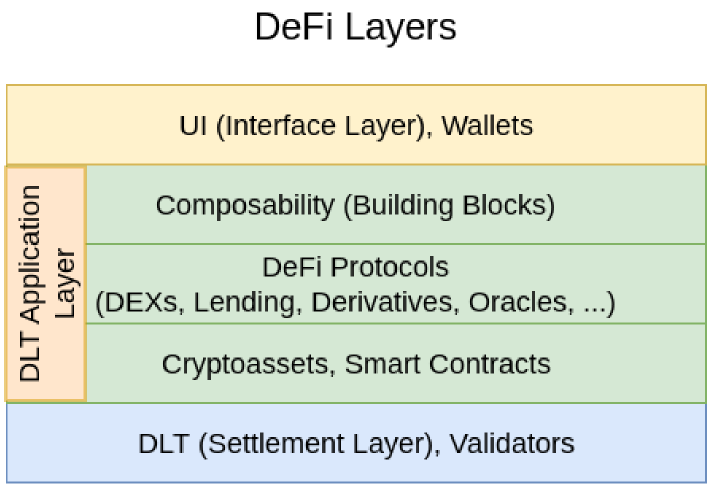

DeFi Layers
UI (Interface Layer), Wallets
DLT Application Layer
Composability (Building Blocks)
DeFi Protocols (DEXs, Lending, Derivatives, Oracles, ...)
Cryptoassets, Smart Contracts
DLT (Settlement Layer), Validators

Figure 3: DeFi Layers

2.3 DeFi Protocols

As with many terms in DeFi, and financial systems more generally, there are no exact or universally agreed definitions for the term DeFi protocol. The agreed concept is that a DeFi protocol is a set of smart contracts, and rules, which govern financial transactions within a decentralized application (dApp) [67], [6], [31].

DeFiLlama.com [1] has extensive listings of DeFi protocols, and in March 2024 more than 3,000 DeFi protocols were listed, and this increases on a daily basis. DEXs are the largest category by a long way (over 1,000 protocols). Other categories are in lending, derivatives (mainly perpetual swaps), yield farming (staking) and more.

2.4 DeFi Primitives

There is no precise or universally accepted definition for the meaning of DeFi primitives. Conceptually, a DeFi primitive is a core component of a DeFi system, but it also means a core component of DeFi interoperability (composition). The literature does not really detail thoroughly this difference [67], [6], [31].

Today, the term DeFi primitive is not only loosely defined, but it has very broad meaning. It can mean core components of a DeFi protocol such as smart contracts, oracles, keepers, governance models, and transactions. But it also can mean core components within a DeFi protocol such as Automated Market Makers (AMMs). Thirdly, it can have a meaning in a more composed context and refer to the actual types of DeFi protocols, such as DEXs, lending [10] [29], staking, RWAs, synthetics, insurance, perpetual swaps [32], payments, bridges, and staking.

Therefore, we have several key concepts which really define how a DeFi protocols works with essential components such as smart contracts, then how it works with what are akin to modules within the DeFi protocol, and then how the entire DeFi protocol works in a composed transaction.

2.5 Automated Market Maker (AMM)

The AMM is a major innovation to DeFi [68], and today AMMs are widely used to process high volumes of trade. The origins of the AMM go back to 2003 with Hanson who produced a scoring formula for market prediction [30]. This concept was taken and developed at Bancor with Jeff Lin [34].

The objective of AMMs is to create continuous liquidity from a liquidity pool. This is done via a conversion function, which originally was a constant product formula, so that the product of two quantities of cryptoassets is constant [11], this is called a Constant Product Market Marker (CPMM). The AMM adjusts price so that one cryptoasset increases in price in the CPMM so that arbitrage is used to balance the quantities by rational actors exchanging more cryptoassets.

Other conversion factors are related to sum (CSMM), mean (CMMM), hybrid (HFMM), and dynamic (DAMM). AMMs are a large subject, and there is a book just about AMMs [62] of 300 pages.

Bartoletti, M et al. [11] formalized some simple AMM ideas using set theory as follows:

We assume an external oracle that provides prices for atomic tokens (i.e. unique type of token). This oracle is modeled as a function P mapping from the set of atomic token types T_0 to the set of non-negative real numbers $\mathbb{R}_{\geq 0}$, indicating that the prices given by the oracle are constant for the duration of the execution:

$$P : T_0 \rightarrow \mathbb{R}_{\geq 0}$$

For a minted token which is represented as an unordered pair of atomic token types $\{\tau_0, \tau_1\}$, the price $P_{\{\tau_0,\tau_1\}}$ depends on the reserves r_0 and r_1 of the atomic tokens τ_0 and τ_1 in the AMM, and the total supply $S_{\{\tau_0,\tau_1\}}$ of the minted token. The price is calculated using the following equation:

$$P_{\{\tau_0,\tau_1\}} = \frac{r_0 \cdot P_{\tau_0} + r_1 \cdot P_{\tau_1}}{S_{\{\tau_0,\tau_1\}}}$$

This equation signifies that the price of a liquidity pool share (minted token) is the weighted average of the underlying atomic tokens' reserves multiplied by their prices, divided by the total supply of the minted token in circulation.

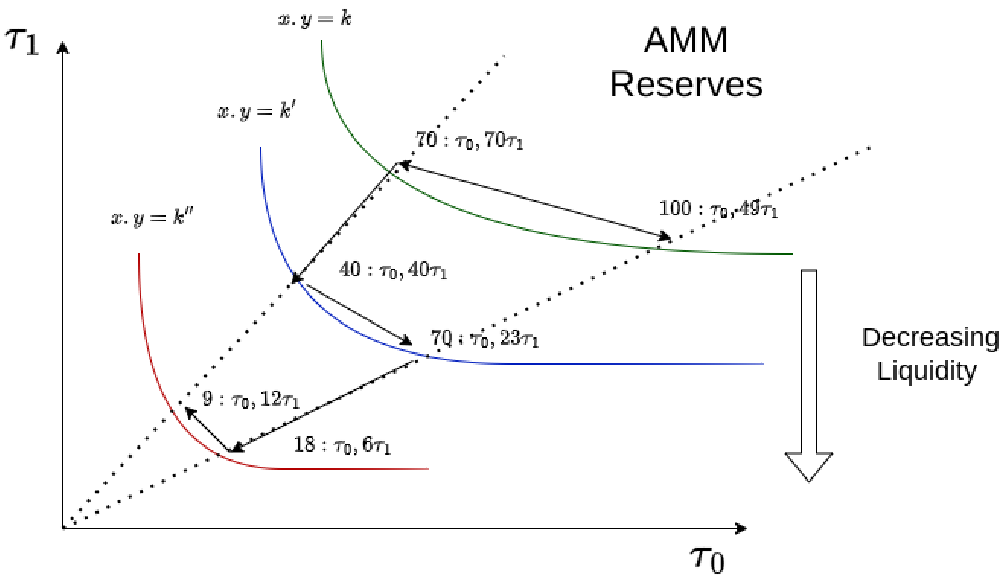


Figure 4: AMM and reserves

2.6 Arbitrage

A key concept in all financial systems, and DeFi, is arbitrage. Arbitrage is a very large subject in financial economics and dozens or even hundreds of pages describe it [64], [38].

The most simple concept of arbitrage is that an asset is trades on two markets at the same time, but the prices differ. Hence a rational actor can buy on one market and sell on another to exploit the price difference and hence gain from the arbitrage.

In DeFi, arbitrage is the basis of a lot of systems. For example in AMMs if the CPMM price on a decentralized exchange (DEX) is different to the market price, then the trade will be adjusted to align the DEX price to the market one. This however also will cause a losee to the trader via slippage.

Similarly in futures, as the maturity date draws close, arbitrage is used to adjust the futures price to the market price.

2.7 DeFi Composition

2.7.1 Introduction

DeFi Composition extends the ideas of smart contract composition (SCC). In SCC, one SC calls another so that one EOA initiated transaction can be completed. In DeFi Composition, the SCC is a a primitive and can be used to build calls between DeFi protocols.

2.7.2 Ethereum Example

In the example shown, there are 3 wallets (EOA 1..3) and 5 SCs (SC A..E). The diagram shows SCC between A $\rightarrow$ B, A $\rightarrow$ C, D $\rightarrow$ E, and then A $\rightarrow$ D is the interaction between the two DeFi protocols and hence we have a composed transaction: EOA1 $\rightarrow$ RWA Protocol $\rightarrow$ Derivatives Protocol. This is called the execution path.

The extra EOA wallets could be used by different users to sign or confirm actions in the protocols (e.g. forming a derivatives contract).

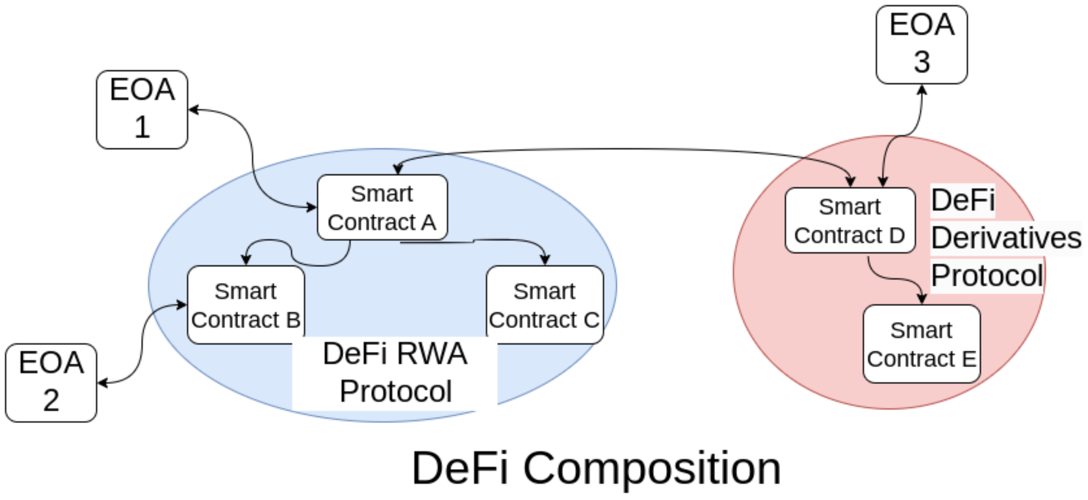


Figure 5: DeFi Composition

2.7.3 DeFi Primitives & Composition

DeFi protocols can then be composed into complex transactions to form more complex financial products. These protocols are primitives, and hence we can assemble an series of calls between primitives. In this example, 5 DeFi primitives (A..E) all form one composed transaction: DeFi Primitives in a Composed DeFi Transaction: EOA $\rightarrow$ A $\leftrightarrow$ B, B $\rightarrow$ C, B $\rightarrow$ D, B $\rightarrow$ D, D $\rightarrow$ E.

DeFi primitives are also called building blocks, money legos, or financial legos.

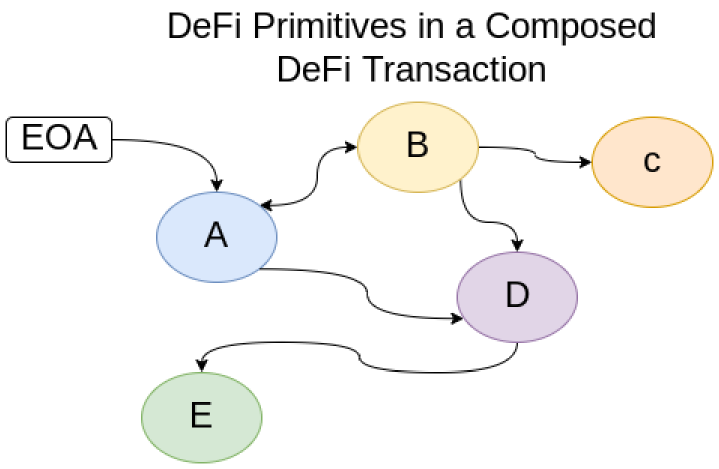


Figure 6: DeFi Composition

Graph Theory

Hence we have graph theory problem. The DeFi protocols are nodes and the execution path links are edges in the graph. There is a cost for execution (e.g. performance, which could be a liquidity value), and hence to optimise cost reduction we could say the following:

Let $D = \{d_1, d_2, \dots, d_{10}\}$ be the set of DeFi protocols represented as nodes in a graph.

We define a set of directed edges E representing possible direct transactions between protocols. For instance, a direct transaction from d_1 to d_3 is an edge $(d_1, d_3) \in E$.

A path P is a sequence of nodes such that each consecutive pair of nodes is connected by an edge in E. For example, a path from d_1 to d_3, then from d_3 to d_7 is represented as $P = (d_1, d_3, d_7)$.

The cost of executing a transaction along an edge is given by a cost function $C : E \rightarrow \mathbb{R}_{\geq 0}$. The total cost of a path is the sum of the costs of its constituent edges.

Our objective is to find a path P^* that minimizes the total cost of execution from a starting node to a destination node, formally defined as:

$$P^* = \arg\min_{P \in \mathcal{P}} \sum_{(d_i, d_j) \in P} C(d_i, d_j)$$

where $\mathcal{P}$ is the set of all possible paths in the graph from the start node to the destination node.

Optimal Paths

Consider a DeFi ecosystem modeled as a weighted directed graph $G = (D, E)$ where:

- $D = \{d_1, d_2, \dots, d_{10}\}$ represents the set of DeFi protocols as nodes.
- E represents the set of direct transactions between these protocols as edges.
- Each edge $(d_i, d_j) \in E$ has an associated cost c_{ij}, representing the transaction cost from d_i to d_j.

The costs are subject to dynamic changes due to factors such as varying interest rates and competitive interactions. Additionally, each node d_i has a liquidity constraint l_i which may limit the transaction capacity.

The objective is to determine an optimal path P^* from a starting protocol d_s to an ending protocol d_t that minimizes the total cost, while respecting the liquidity constraints. Formally, we seek:

$$P^* = \arg\min_{P \in \mathcal{P}} \sum_{(d_i, d_j) \in P} c_{ij}$$

where $\mathcal{P}$ is the set of all paths from d_s to d_t in the graph G, and c_{ij} are the dynamically adjusted costs.

A feedback mechanism is incorporated to update the graph G with the latest cost and liquidity information. A suitable optimization algorithm, e.g. Dijkstra's shortest path algorithm [37] is employed to continually find the most cost-effective path in this changing environment.

Paths

The graph problem quickly gets very complicated as follows. Consider a graph with 3 nodes, where we start from a specific node, say d_1, and want to find all possible paths that can include any subset of the remaining nodes. For 3 nodes, we have:

- Direct paths from d_1 to each of the other nodes: 2 paths.
- Paths that include 1 additional node, making a 2-step path: $2 \times 1 = 2$ paths, since after choosing the first step, only 1 option remains for the second step.

Thus, the total number of paths for a 3-node graph starting from d_1 is $2 + 2 = 4$.

General Formula for n-Node Graph

For a graph with n nodes, starting from a specific node and considering paths of varying lengths, the total number of paths can be calculated using the formula:

$$\text{Total Paths} = \sum_{k=1}^{n-1} \frac{(n-1)!}{(n-1-k)!}$$

Where:

- k represents the length of the path, ranging from 1 to $n-1$.
- The term $\frac{(n-1)!}{(n-1-k)!}$ calculates the number of permutations of $n-1$ nodes taken k at a time, representing the different paths of length k that can be formed from the remaining $n-1$ nodes.

This formula accounts for all paths starting from the specified node, with lengths varying from 1 to $n-1$, without revisiting any node.

For 10 nodes, over 3.6 million paths exist, and for the number of DeFi protocols which exist now (3500), over 10^{80} paths exist (more than the number of atoms that exist in the universe).

2.8 Router

The Router SC is typically the SC which receives the message from the EOA. The router will make decisions about which building blocks to call to meet the wallet requirement.

2.8.1 Composition Graphs and Trees

Kitzler et al. [40] did extensive research analysing more than 10 million EOAs. They produced visualizations of DeFi compositions as executive trees and squares representing building blocks. They also concluded that most activity was with DEXs and lending for DeFi composition and very little for derivatives. They also suggested a sidechain could be used for derivatives transactions. This is probably because derivatives trade tends to be very high volume as financial contracts can be traded.

2.9 DeFi Security and Compositional Security

2.9.1 Introduction

DeFi systems have been attacked numerous times causing large losses (billions of dollars). Li [44] published a good overview of the attacks in DeFi categorizing them as attacks on oracles, wallets, smart contracts, flash loans, and blockchains (e.g. transaction ordering, and forking). The large number of successul attacks has revealed there is considerable work needed in the DeFi sector for security improvement. Smart contract vulnerabilities are the cause of a lot of losses.

If an individual DeFi primitive is exploited when composed (i.e. a building block) then the entire composed transaction is exploited. Therefore, a vulnerability in one DeFi primitive can cascade through an entire transaction flow causing heavy losses.

Due to the large number of attacks, and the size of the security aspects in DeFi, we include a short summary only in this document of some attack vectors.

2.9.2 Oracles

Oracles have been successfully attacked for example when supplying market prices to AMMs, Wang et al. detailed several attacks in 2020 on Warp Finance, Value DeFi, Cheese Bank, Plouto Finance, Harvest Finance, and bZx [65]. As with flash loan attacks, the attacker of an oracle can complete numerous steps in one transaction, this is done by distorting prices in the oracle price feeds which are then exploited on an DEX.

2.9.3 Wallets

Wallet attacks usually involve an attacker stealing the private keys of the wallet. Private key leakage has been observed in software and hardware wallets [27].

2.9.4 Transaction Ordering

These attacks manipulate the sequence of transactions as the blocks are created, such as frontrunning, backrunning, and sandwich attacks.

Frontrunning is an attack where a malicious node observes a transaction after it is broadcast but before it is finalized, and attempts to have its own transaction confirmed before or instead of the observed transaction [22].

Backrunning is similar to frontrunning—the knowledge of a transaction is used to insert an order right after the target transaction. In most cases, the price slippage due to a transaction will change the exchange rate on one exchange, but not on others. Using this information, backrunners exploit the price difference between multiple DEX exchanges to turn a profit. In this work, backrunning extracts miner extractable value (MEV), the profit to the miner, through a single transaction in which the same cryptocurrency is exchanged on multiple exchanges [50].

The sandwich attack is when attackers place their transactions around a victim's order, capitalizing on price slippage [45].

2.9.5 Forking

This attack is when a node is incentized to change past blocks and this create a fork. This attack is called a time-bandit attack [54] [41].

2.9.6 Smart Contracts

The subject of smart contract vulnerabilities is a very large subject area and has been written about ever since smart contracts were introduced. Because DeFi protocols are made up of smart contracts then when the SC can be exploited then the entire DeFi protocol can be exploited.

We can summarize some main aspects of SC vulnerabilities but a fuller text would devote hundreds of pages to this task [18].

Suicidal and Greedy Contracts	**Block Info Dependency**
A suicide function designed to end a contract, can be misused by an attack to unintentionally kill a contract. A greedy contract locks funds (hence greedy) and there is no function to withdraw.	Timestamps can vary between blocks but if the timestamp is used in the application, then it can be exploited.
Unchecked External Call	**Reentrancy Attack**
If the value of an external call is not checked, an attacker can exploit the fact if the external function fails the calling contract still executes.	This is a common attack in which program flow is adjusted to adversely adjust a contract state [16].
Arithmetic Bug	
Attackers exploit vulnerabilities allowing underflow and overflow in the smart contract.	

Table 1: Common Smart Contract Vulnerabilities

2.10 Flash Loans

Flash loans are a novel concept, invented by Marble in 2018 (now Aave [25]) which allow a borrower to borrow without any collateral who then repays the loan in the same blockchain transaction. This allows a borrower to borrow almost any amount (e.g. millions of dollars) without collateral.

But flash loans have been misused, e.g. via a pump attack. In this attack, large volumes of money are used to distort prices on a DEX (using AMMs) which are then exploited by the attacker by engaging in numerous transactions via composition so that price distortions can be used for gain [15] [51]. The precise details would take hundreds of words to explain, and hence a summary is given.

2.11 Vulnerability Detection

Formalism has been researched by Tolmach et al. [61] using process algebra methods by considering Communicating Sequential Process and modelling. Process algebra models a system [7].

Cecchetti, Ethan et al. examined reentrancy attacks via a formal framework leveraging information flow control (IFC) techniques [16].

These approaches by Tolmach et al. and Cecchetti, Ethan et al. are not widely adopted. The methods to solve vulnerabilities in DeFi are mainly via bug fixes following attacks, offering bounties to ethical hackers, audits of smart contracts, and using well tested libraries.

IV. Optimization

3 Optimization Approaches

3.1 Introduction

We consider a key question to ask is: What are the effects of liquidity aggregation mechanisms on the capital efficiency and slippage in ComFi markets?

And: How can the efficacy of risk mitigation mechanisms in ComFi be quantified and optimized to ensure the secure creation, trading, and settlement of complex derivative contracts on tokenized real-world assets?

These questions lead into the concept of the Aquo product which is to optimize DeFi and actually the concept extends to any AMM and liquidity based financial system of which DeFi is a use case.

3.2 Solution Approaches

The AMM is the cornerstone to DeFi trades and optimizing AMMs is a key objective. There are several approaches which could be taken as follows:

1. Reinforcement Learning (RL)
2. Genetic Algorithms (GA)
3. Simulated Annealing (SA)
4. Particle Swarm Optimization (PSO)
5. Lagrangian Multipliers (Optimal Control)
6. Dynamic Programming (DP)
7. Integer Linear Programming (ILP)
8. Convex Optimization

3.3 Approaches

We consider that the best approach is via reinforcement learning, but we will provide an overall of Convex Optimization as that is widely used in solving optimization related problems.

3.4 Convex Optimization

In considering this subject, we can consider the AMM at first.

The subject matter related to AMMs is rooted in a lot of theory about automated market markets which emerged in 2003 and was continued with a number of papers on cost function market makers.

Chen et al. made a number of contributions eg [3] and analysed how prediction markets affected automated market markers. These publications pre-date Ethereum and reflect how DeFi emerged from a lot of theory which arose in computational economics and mathematical finance.

4 AMMs

We consider as the basic building block in a DeFi Protocol the AMM. We consider an AMM where $xy = k$ and determine the relationship between Δ_y and Δ_X and determine the following:

Given that $yx = k$ where x and y are initial values,consider a directional change in x to be Δx.

Then, $yx = k$

$\Rightarrow (y + \Delta y)(x + \Delta x) = k$, where Δy is the change in y.

$\Rightarrow yx + y\Delta x + x\Delta y + \Delta y\Delta x = k$

$\Rightarrow y\Delta x + x\Delta y + \Delta y\Delta x = 0$ (since $yx = k$)

$\Rightarrow \Delta y(x + \Delta x) = -y\Delta x$

$\Rightarrow \Delta y = -\dfrac{y\Delta x}{x + \Delta x}$.

$$\Delta y = -\frac{\left(\frac{k}{x}\right)\Delta x}{x + \Delta x}\Delta y = -\frac{k\Delta x}{x^2 + x\Delta x}\Delta y = -\frac{k\Delta x}{x(x + \Delta x)}$$

This is highly non-linear and multivariable.

We can visualize this problem definition via the bonding curves shown in 7. They show different returns of Δy for different Δx, k, and x (initial value).

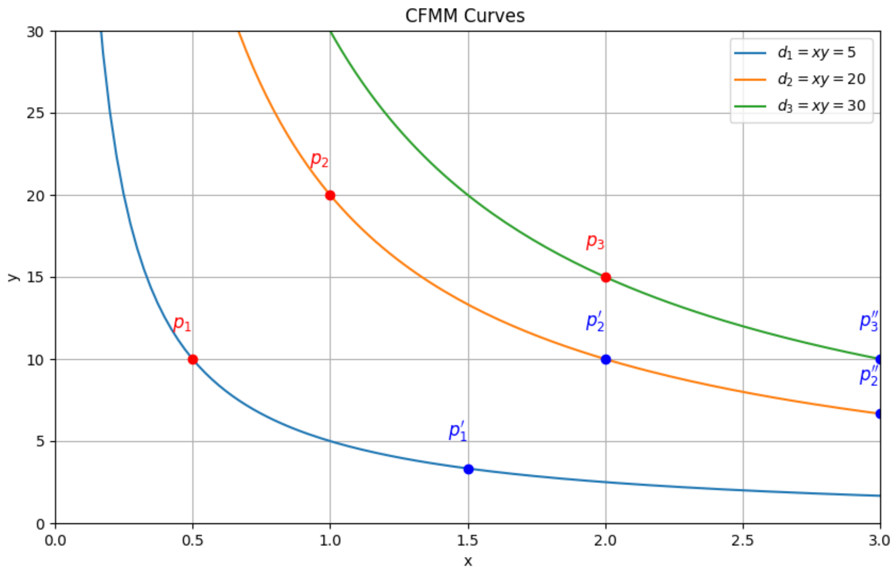


Figure 7: Bonding Curves

Optimizing Δy

It is clear from the expression for a DeFi Protocol that Δy will vary according to x (current value on the bonding curve), Δx, and k. We consider k to be constant, ie we ignore concentrated liquidity. Then it is self-evident that if Δx can be split then quantity of Δy will change. Therefore, if we have multiple DeFi Protocols, we can split the input Δx to optimize the output Δy.

We consider n DeFi Protocols.

We consider a transaction which sends Δx to be a set of DeFi Protocols (DEXs) which return Δy. The objective is maximise Δy. We also normalise the input so that we can more easily measure results and we can always just scale by a constant factor. We consider for example Δx is 1 Eth and Δy is the USDC returned.

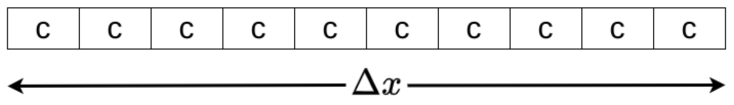


Figure 8: Δx Split

Then we can construct blocks of c to send to the DeFi Protocols as follows: $S = \{0, c, 2c, \ldots, 1\}$ where k divides 1 exactly. This is a normalized set of inputs. Hence if Δx is sent it is a combination of S elements. Hence we have $m = \frac{1}{c} + 1$ elements in S.

Taking an example, of $c = 0.1$ we have $\{0, 0.1, 0.2...1\}$ then for 3 DeFi Protocols we have $\{0, 0, 1\}, \{0, 0.1, 0.9\}$ and 66 combinations. Hence m is 11 in this example.

We consider the actual inputs to the DeFi Protocols $\Delta_x = \{\Delta x_1, \Delta x_2, \ldots, \Delta x_n\}$ such that $\sum \Delta x_i$ = 1 (normalized) and each $\Delta x_i = cq_i$ where q_i is an integer. The size of each subset can be up to n, where $n \leq m$. Therefore:

$$q_1 + q_2 + \ldots + q_m = \frac{1}{c}$$

We can visualize this as follows:

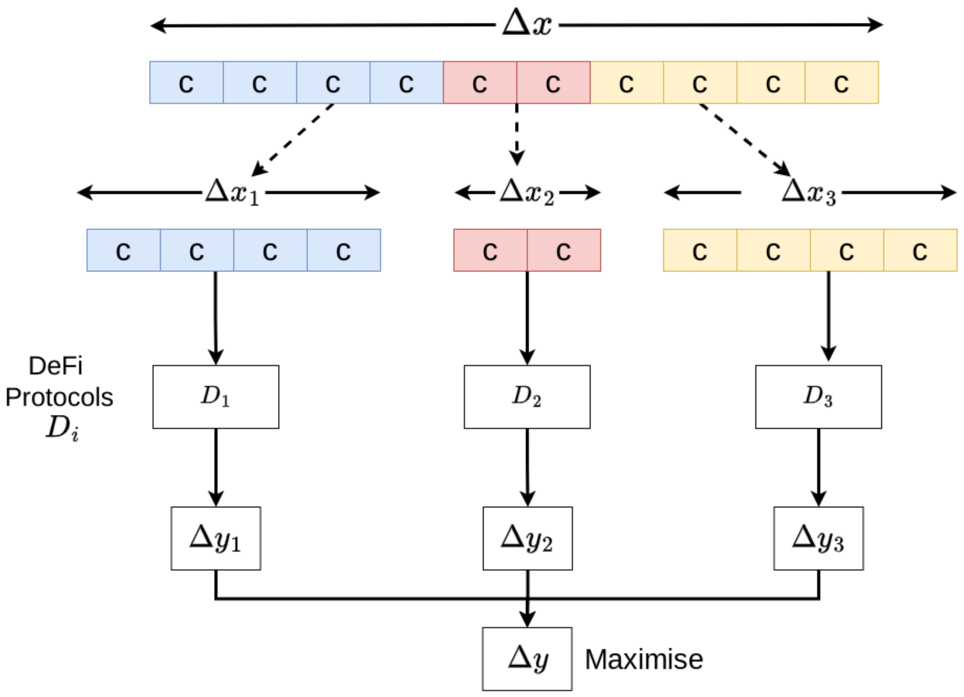


Figure 9: Δ_x & Δ_y

Application of Stars and Bars Theorem

This problem now is to construct the distribution of Δx_i values. We can consider this in terms of Combinatorics [4]. We have

Using the "stars and bars" theorem, the number of ways to distribute $\frac{1}{c}$ identical units into n parts is given by the binomial coefficient:

$$\binom{\frac{1}{c}+n-1}{n-1}$$

Factorial Representation

Expanding this in terms of factorials, we have for number of possible inputs to the DeFi Protocols:

$$N = \binom{\frac{1}{c}+n-1}{n-1} = \frac{(\frac{1}{c}+n-1)!}{(n-1)! \times (\frac{1}{c})!}$$

We can consider a matrix of all possible inputs:

For m iterations and n DeFi protocols, we have:

$$\Delta X_{n\times N} = \begin{bmatrix} \Delta x_{11} & \Delta x_{12} & \cdots & \Delta x_{1n} \\ \Delta x_{21} & \Delta x_{22} & \cdots & \Delta x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta x_{N1} & \Delta x_{N2} & \cdots & \Delta x_{Nn} \end{bmatrix}$$

Which produce a series of Δy values as follows:

$$\Delta Y_{n\times N} = \begin{bmatrix} \Delta y_{11} & \Delta y_{12} & \cdots & \Delta y_{1n} \\ \Delta y_{21} & \Delta y_{22} & \cdots & \Delta y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta y_{N1} & \Delta y_{N2} & \cdots & \Delta y_{Nn} \end{bmatrix}$$

Where the row is for the n DeFi Protocols. Hence we sum these to get a column matrix of aggregated Δy values.

$$\Delta y_a = \begin{pmatrix} \sum_j^n \Delta y_{1j} \\ \sum_j^n \Delta y_{2j} \\ \vdots \\ \sum_j^n \Delta y_{Nj} \end{pmatrix}$$

We then take the optimal value in Δy_a.

Hence we can state the problem statement:

To optimize Δy for an amount Δx by taking into account initial prices at DEXs and k values. This is a symetrical problem when considering $xy = k$ an therefore of Δy is deposited into the exchange the the solution is the same as for Δx being deposited.

4.1 Building Blocks

Building Blocks or money legos are widely referenced in the literature but without much detail about how a block is defined. We now have more of a criteria to define a building block.

We know from the preceding analysis, that each protocol performs according to x (current quantity) and Δx, and k. Therefore we can define a building block in these terms. Specifically a building block can be characterized by the states which affect optimization of Δy. Hence we can say:

Building Block $f_i(x, y) = f_i(k_i, x_{i0}, \Delta x_i, g_i(x, y), \Delta y_i)$ where $g_i(x, y)$ is the bonding curve, eg $xy = k$, and in the more specific case of $xy = k$, Δy is a function of the other variables.

Therefore to combine or compose these blocks into a transaction, we apply this standard, so we seek to build a function:

$$f(x, y) = \sum w_i f_i(x, y)$$

where $w_i \geq 0$

5 Testing

To test out the basics of accessing AMMs and determining the efficiency levels, we consider three AMMs. We consider trading 1ETH for USDC at different exchanges. The objective is to get as much USDC as possible.

Each AMM has an initial price which is its place on the bonding curve. Each AMM is assumed to $xy = k$ with varying k values.

Port	Initial X (ETH)	Initial Y (USDC)	k
3008	8	31,200	249,600
3009	10	41,000	410,000
3010	11	41,000	451,000

Table 2: Data

We can visualize this as follows:

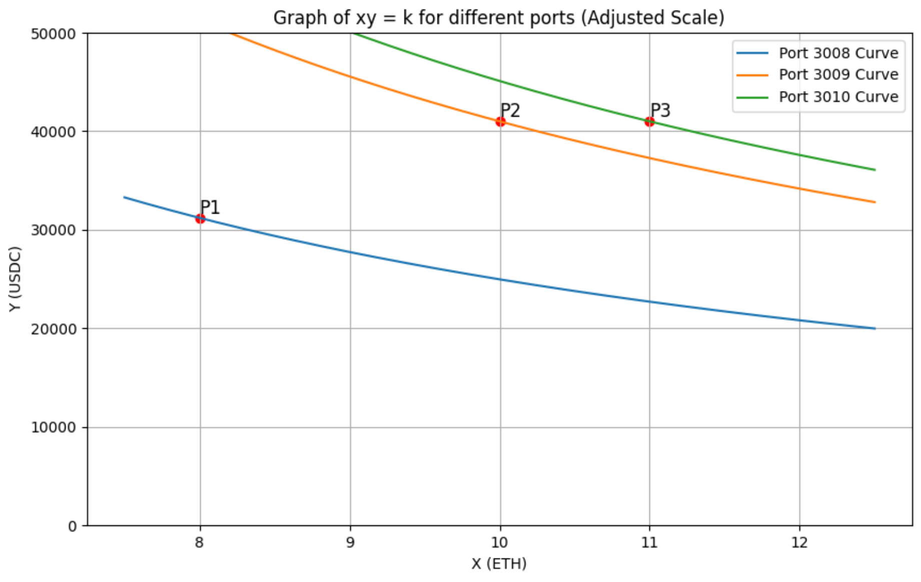


Figure 10: Test Bonding Curves

We then send data to the 3 AMMs in normalized form for 1 ETH. And record the outcomes. We have 0.1 increments, and hence the table 3 is an example.

We have 66 segments in total. We represent this as follows. A vector V has m rows for all combinations to split the asset to trade. We take 1 ETH and split into 0.1 increments, hence we have 66 possible trades.

5.1 Results

We test and find the results below. These show optimal trade happened when ETH was split $0.3, 0.6, 0.1$ and that gave a normalized value 1. The worst trade was normalized at 0.895. Assuming an Ether price of 4,000 USDC, that makes a loss of 419 USDC per Ether without optimization.

```
{
    _id: ObjectId('661b96e4d10db03a5198bf26'),
    path: 37,
    x: 0.3,
    y: 0.6,
```

x	y	z
0.3	0.4	0.3
0.3	0.5	0.2
0.3	0.6	0.1
0.3	0.7	0.0
0.4	0.0	0.6
0.4	0.1	0.5

Table 3: Data

```
    z: 0.1,
    x_delta: 1,
    y_delta: 1,
    slippage: 794.703
}
```

Listing 2: MongoDB Document

```
{
{
    _id: ObjectId('661b96e4d10db03a5198bf02'),
    path: 1,
    x: 0,
    y: 0,
    z: 1,
    x_delta: -1,
    y_delta: 0.895,
    slippage: 595.328
  }
```

Listing 3: MongoDB Document

In the graph 11 the yellow dots show optimal trades.

Plot Delta Y, X, Y, Z

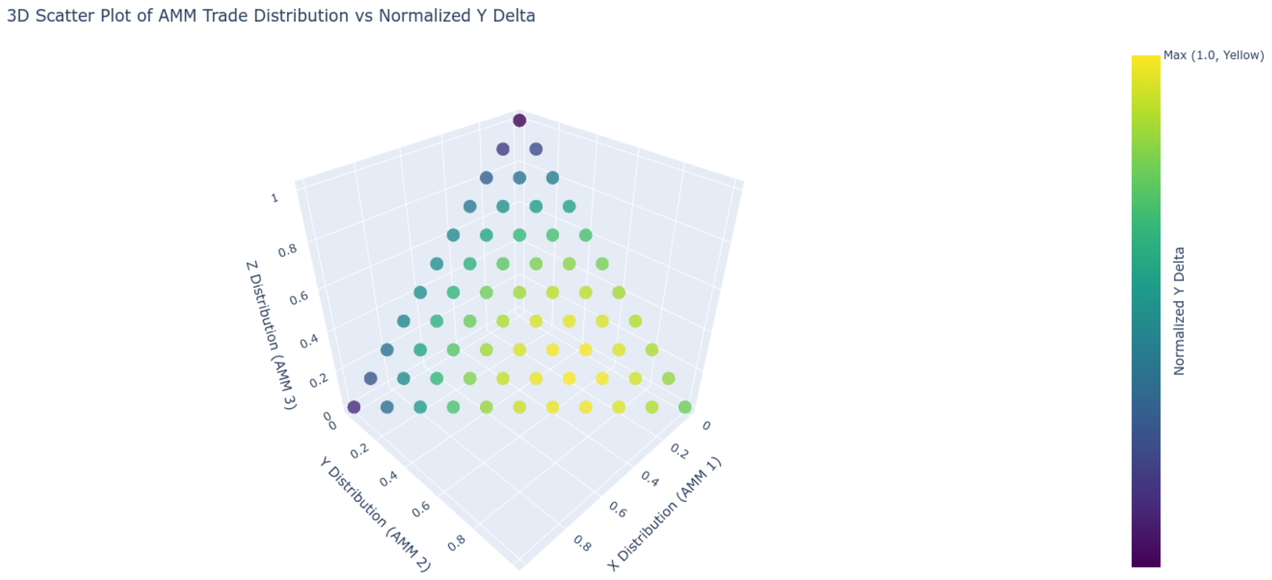


Figure 11: Test Results

Plot Delta Y Values

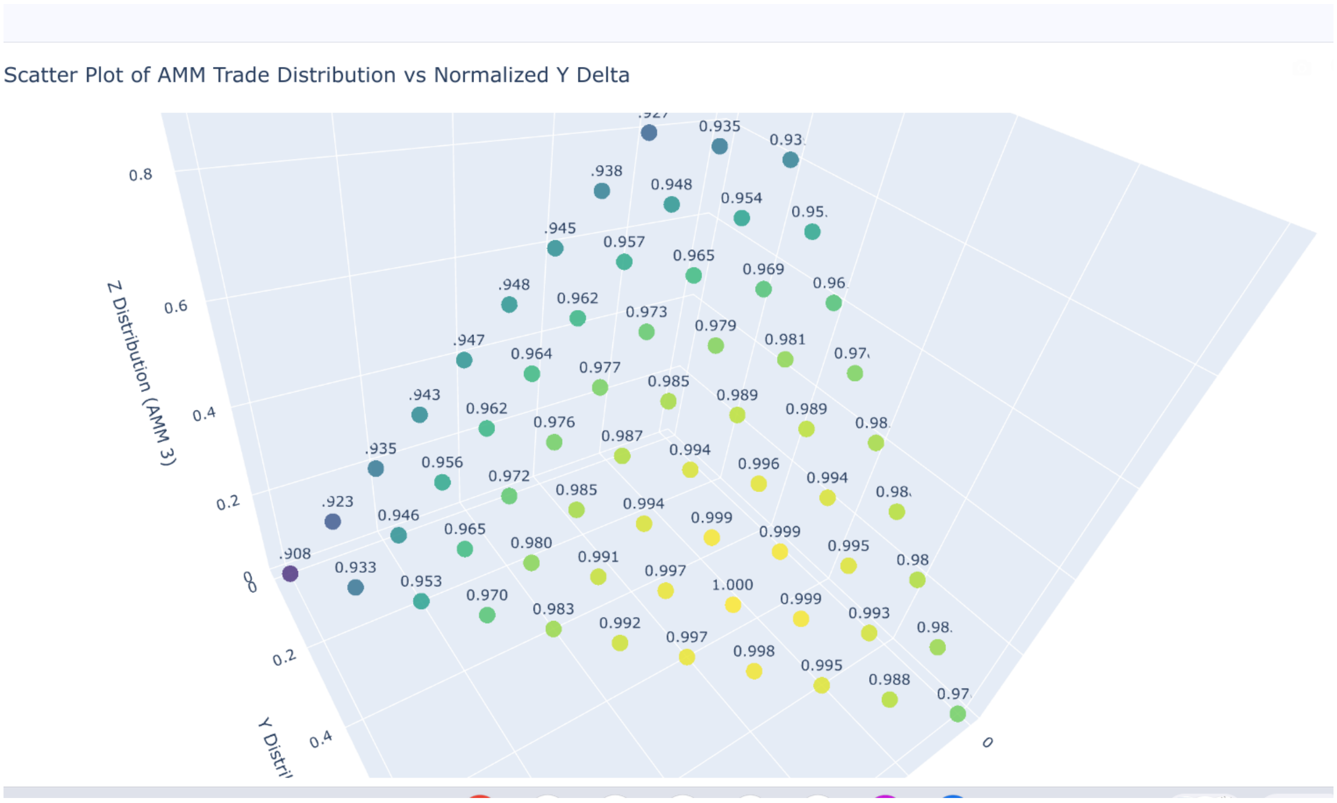


Figure 12: Plot Labels

Plot Delta Y Fixed Z

We can consider z=0.1 and then see the non-linear nature of the distribution.

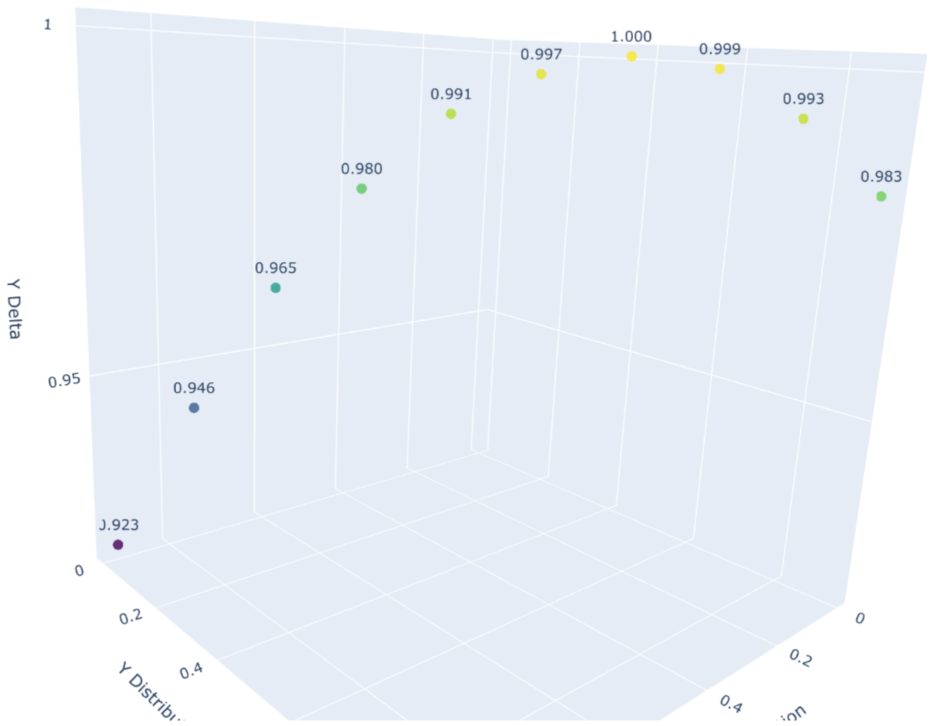


Figure 13: Fixed Z Value

V. Reinforcement Learning

6 Reinforcement Learning (RL)

6.1 Introduction

We consider RL is a viable approach to solve these optimization problems and we outline some key preliminary matters.

6.2 Preliminaries

The agent sends an action and reads an observation. Then a reward is calculated. For the AMM use case, the action is a ΔX transaction send, and the observation is Δy. The reward would be 1.0 for a zero slippage AMM trade and otherwise reduced by an amount normalized across the breadth of values (e.g. if the worst trade is 10% less then we have values 1.0 to 0.9 or we can normalize the actual range ie 1.0 to 0 with 0 being 10%).

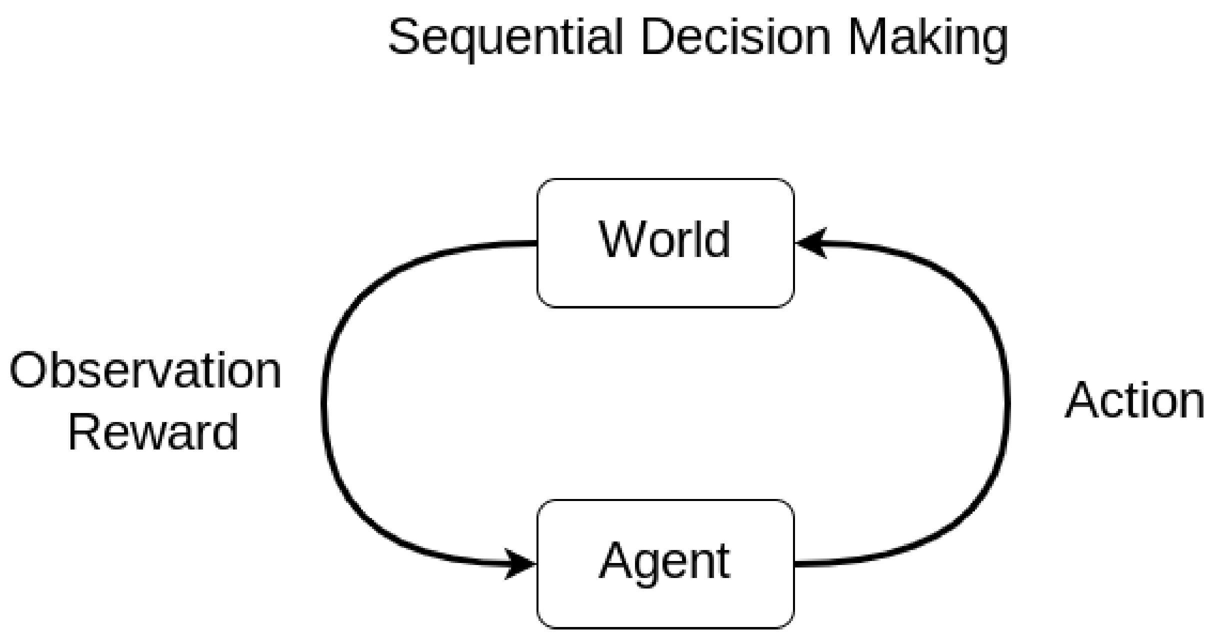


Figure 14: Sequential Decision Making

Possible solutions as as follows:

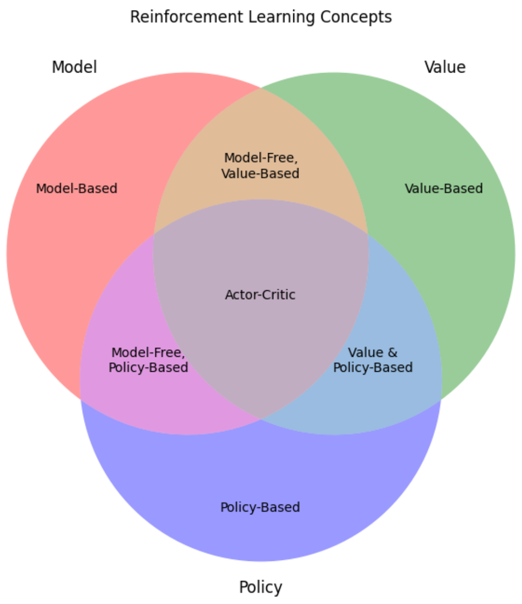


Figure 15: Solution Types

RL sits between supervised learning and unsupervised learning; there is sequential decision making and limited feedback.

There are two classes of algorithms, RL and dynamic programming [63]. We describe some useful terms.

6.3 State

We have two states, environment and agent state. For time t, and History H we have:

$$S_t = f(H_t)$$

(environment state)

$$S_t^a = f(H_t)$$

(agent state)

Markov state:

$$P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \ldots) = P(S_{t+1}|S_t, A_t) = T(S_t, A_t, S_{t+1})$$

This says S_{t+1} in an MDP (Markov Decision Process) depends only on the current state S_t and the current action A_t, and jot the history of states.

The future is independent of the past. The state is a sufficient statistic of the future.

Hence we have three states in total.

Fully observable, the agent directly observes the environment state:

$$O_t = S_t^a = S_t^e$$

This concludes that the agent state = environment state = information state, this is a Markov Decision Process (MDP).

Partial observability, agent indirectly observes the environment. The agent state is not the environment state.

RNN - Recurrent Neural Network.

6.4 Policy

We can consider policy to be the agent's behaviour.

We can consider different policies for RL.

Deterministic Policy

A **deterministic policy** is a function $\pi : \mathcal{S} \to \mathcal{A}$, which maps a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. For any given state s, a deterministic policy prescribes a specific action to be taken.

We can write this as $a = \pi(s)$

Stochastic Policy

A **stochastic policy** is a function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which gives a probability distribution over actions for each state. In this case, for a given state s, the policy π outputs a probability $\pi(a|s)$ that the agent takes action a.

In short, this is about a probability,

$$\pi(a|s) = P[A = a|S = s]$$

This means for a specific action (a) from A, then it is conditional on a specific s from S.

The goal in many RL scenarios is to find an **optimal policy** π^* that maximizes the expected cumulative reward over time, **starting from any given state**. This involves evaluating the expected return for policies and improving them iteratively, typically using algorithms like Policy Iteration or Value Iteration.

6.5 Value

This is defined by how good is each state and/or action.

This is a prediction of expected future reward as follows [66]:

$$V(s) = \mathbb{E}_{s,a\sim\pi}\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\right]$$

Here, $V(s)$ is the value function of state s, $\mathbb{E}_{s,a\sim\pi}$ denotes the expected value given that states and actions follow policy π, γ is the discount factor, and R_{t+k+1} are the rewards received at future time steps.

$$V(s) = \sum_{s',a}\pi(a|s)P(s_{t+1}=s'|s_t=s,a_t=a)\left(R_{t+1}+\gamma\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}\right)$$

In the detailed Bellman equation, $\pi(a|s)$ is the probability of taking action a in state s under policy π, $P(s_{t+1}=s'|s_t=s,a_t=a)$ is the probability of transitioning to state s' from state s after taking action a, and R_{t+1} is the immediate reward for that transition.

$$V_\pi(s) = \mathbb{E}_\pi\left[R_{t+1}+\gamma V(S_{t+1})|S_t=s\right]$$

The simplified Bellman equation represents the value of state s as the expected immediate reward R_{t+1} plus the discounted value of the subsequent state, all conditioned on the current state being s.

6.6 Model

This is the agent's representation of the environment. There are two models, Transition and Reward.

For a Transition model, P predicts the next state (i.e. dynamics).

For the Reward model, the next (immediate) reward is predicted.

State transition model:

$$P^a_{ss'} = P(S'=s' \mid S=s, A=a)$$

$$R^a_S = E[R|S=s,A=a]$$

6.7 Solutions

Models are not needed and we can have model-free solutions.

We have also policy based solutions, policy based agents which store policies. Model solutions store the model. An active critic is a combination of both.

7 Test Results

The work is on-going to provide RL test results.

VI. Business Model

8 Business

The business model is rooted around a AI network being able to optimize DeFi trades.

8.1 Users

The Aquo model is a decentralized model and users will have control over their own finances via wallets. Aquo does not propose any centralized holding of funds (no custody).

A user can be a retail consumer or a business.

8.2 Fees

A transaction fee would be charged inline with many business models in the DeFi space.

8.3 UX

The user will initiate a transaction via Aquo on an UI and Aquo then will execute it using DeFi Composition.

8.4 UVP

The UVP is the optimization aspect which cannot easily be copied because the user benefits from learned behaviour in the RL system.

References

[1] Defi llama. https://defillama.com/. Accessed on March 25, 2024.

[2] The sveriges riksbank prize in economic sciences in memory of alfred nobel 1997. https://www.nobelprize.org/prizes/economic-sciences/1997/press-release/, 1997.

[3] Jacob Abernethy, Yiling Chen, and Jennifer Wortman Vaughan. An optimization-based framework for automated market-making. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 297–306, 2011.

[4] Henry Adams, Kelly Emmrich, Maria Gillespie, Shannon Golden, and Rachel Pries. Counting rocks! an introduction to combinatorics. *arXiv preprint arXiv:2108.04902*, 2021.

[5] Nicholas Apostolou and D Larry Crumbley. The tally stick: the first internal control. *The Forensic Examiner*, 70(1):60–62, 2008.

[6] Raphael Auer, Bernhard Haslhofer, Stefan Kitzler, Pietro Saggese, and Friedhelm Victor. The technology of decentralized finance (defi). *Digital Finance*, pages 1–41, 2023.

[7] Jos CM Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, 2005.

[8] Bank of England. Money creation in the modern economy, 2014.

[9] Bank of England. Quarterly bulletin 2014 q1, 2014. Accessed: 2024-03-23.

[10] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch Lafuente. Sok: lending pools in decentralized finance. In *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC, Virtual Event, March 5, 2021, Revised Selected Papers 25*, pages 553–578. Springer, 2021.

[11] Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of automated market makers in defi. *Logical Methods in Computer Science*, 18, 2022.

[12] Zvi Bodie and Alex Kane. Investments. 2020.

[13] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 1:1–136, 2021.

[14] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

[15] Yixin Cao, Chuanwei Zou, and Xianfeng Cheng. Flashot: a snapshot of flash loan attack on defi ecosystem. *arXiv preprint arXiv:2102.00626*, 2021.

[16] Ethan Cecchetti, Siqiu Yao, Haobin Ni, and Andrew C Myers. Compositional security for reentrant applications. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1249–1267. IEEE, 2021.

[17] Varadarajan V Chari, Lawrence Christiano, Patrick J Kehoe, et al. Facts and myths about the financial crisis of 2008. *Federal Reserve Bank of Minneapolis Working Paper*, 666, 2008.

[18] Jiachi Chen, Xin Xia, David Lo, John Grundy, Xiapu Luo, and Ting Chen. Defining smart contract defects on ethereum. *IEEE Transactions on Software Engineering*, 48(1):327–345, 2020.

[19] Poulami Das, Andreas Erwig, Sebastian Faust, Julian Loss, and Siavash Riahi. The exact security of bip32 wallets. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 1020–1042, 2021.

[20] Torsten Ehlers and Bryan Hardy. The evolution of otc interest rate derivatives markets. *BIS Quarterly Review, December*, 2019.

[21] Nabil El Ioini and Claus Pahl. A review of distributed ledger technologies. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II*, pages 277–288. Springer, 2018.

[22] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 170–189. Springer, 2020.

[23] Frank J Fabozzi and Moorad Choudhry. The handbook of european structured financial products. 2004.

[24] Guido Ferrarini and Paolo Saguato. Regulating financial market infrastructures. *Forthcoming draft chapter, The Oxford Handbook on Financial Regulation, edited by Eilís Ferran, Niamh Moloney, and Jennifer Payne,(Oxford University Press)., European Corporate Governance Institute (ECGI)-Law Working Paper*, (259), 2014.

[25] Emilio Frangella and Lasse Herskind. Aave v3 technical paper. 2022.

[26] Juan A Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Proofs of work for blockchain protocols. *IACR Cryptol. ePrint Arch.*, 2017:775, 2017.

[27] Andriana Gkaniatsou, Myrto Arapinis, and Aggelos Kiayias. Low-level attacks in bitcoin wallets. In *Information Security: 20th International Conference, ISC 2017, Ho Chi Minh City, Vietnam, November 22-24, 2017, Proceedings 20*, pages 233–253. Springer, 2017.

[28] Jeff Gray and Bernhard Rumpe. Models for digitalization, 2015.

[29] Lewis Gudgeon, Sam Werner, Daniel Perez, and William J Knottenbelt. Defi protocols for loanable funds: Interest rates, liquidity and market efficiency. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 92–112, 2020.

[30] Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5:107–119, 2003.

[31] Campbell R Harvey, Ashwin Ramachandran, and Joey Santoro. *DeFi and the Future of Finance*. John Wiley & Sons, 2021.

[32] Songrun He, Asaf Manela, Omri Ross, and Victor von Wachter. Fundamentals of perpetual futures. *arXiv preprint arXiv:2212.06888*, 2022.

[33] Roger Heines, Christian Dick, Christian Pohle, and Reinhard Jung. The tokenization of everything: Towards a framework for understanding the potentials of tokenized assets. In *PACIS*, page 40, 2021.

[34] Eyal Hertzog, Guy Benartzi, and Galia Benartzi. Bancor protocol. *Continuous Liquidity for Cryptographic Tokens through their Smart Contracts. Available online: https://storage. googleapis. com/website-bancor/2018/04/01ba8253-bancor_protocol_whitepaper_en. pdf (accessed on 6 June 2020)*, 2017.

[35] Claude Humbel. Decentralized finance: A new frontier of global financial markets regulation. *GesKR: Schweizerische Zeitschrift für Gesellschafts-und Kapitalmarktrecht sowie Umstrukturierungen*, (1):9–25, 2022.

[36] Investopedia. Structured finance. https://www.investopedia.com/terms/s/structuredfinance.asp, 2021.

[37] Donald B Johnson. A note on dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20(3):385–388, 1973.

[38] Yuri M Kabanov and Dmitry O Kramkov. Asymptotic arbitrage in large financial markets. *Finance and Stochastics*, 2:143–172, 1998.

[39] Kostis Karantias. Sok: A taxonomy of cryptocurrency wallets. *Cryptology ePrint Archive*, 2020.

[40] Stefan Kitzler, Friedhelm Victor, Pietro Saggese, and Bernhard Haslhofer. Disentangling decentralized finance (defi) compositions. *ACM Transactions on the Web*, 17(2):1–26, 2023.

[41] Lukas König, Stefan Unger, Peter Kieseberg, Simon Tjoa, and Josef Ressel Center Blockchains. The risks of the blockchain a review on current vulnerabilities and attacks. *J. Internet Serv. Inf. Secur.*, 10(3):110–127, 2020.

[42] Steve Kummer and Christian Pauletto. The history of derivatives: A few milestones. In *EFTA Seminar on Regulation of Derivatives Markets*, pages 431–466, 2012.

[43] Yue-Kuen Kwok. *Mathematical models of financial derivatives*. Springer, 2008.

[44] Wenkai Li, Jiuyang Bu, Xiaoqi Li, and Xianyi Chen. Security analysis of defi: Vulnerabilities, attacks and advances. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 488–493. IEEE, 2022.

[45] Yuxuan Liang, Xukang Wang, Ying Cheng Wu, Hongpeng Fu, and Mengjie Zhou. A study on blockchain sandwich attack strategies based on mechanism design game theory. *Electronics*, 12(21):4417, 2023.

[46] Nitima Masla, Vaibhav Vyas, Jyoti Gautam, Rabindra Nath Shaw, and Ankush Ghosh. Reduction in gas cost for blockchain enabled smart contract. In *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 1–6. IEEE, 2021.

[47] Satoshi Nakamoto. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*, 9:15, 2008.

[48] Larry Neal. *A concise history of international finance: From Babylon to Bernanke*. Cambridge University Press, 2015.

[49] Gustavo A Oliva, Ahmed E Hassan, and Zhen Ming Jiang. An exploratory study of smart contracts in the ethereum blockchain platform. *Empirical Software Engineering*, 25:1864–1904, 2020.

[50] Julien Piet, Jaiden Fairoze, and Nicholas Weaver. Extracting godl [sic] from the salt mines: Ethereum miners extracting value. *arXiv preprint arXiv:2203.15930*, 2022.

[51] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In *International conference on financial cryptography and data security*, pages 3–32. Springer, 2021.

[52] Przemysław Rodwald. An analysis of data hidden in bitcoin addresses. In *International Conference on Dependability and Complex Systems*, pages 369–379. Springer, 2021.

[53] Mona L Scott and MONA L SCOTT. Dewey decimal classification. *Libraries Unlimited*, 1998.

[54] Venkkatesh Sekar. *Preventing front-running attacks using timelock encryption*. PhD thesis, Ph. D. thesis, University College London, 2022.

[55] Avi Silberschatz, Michael Stonebraker, and Jeff Ullman. Database systems: Achievements and opportunities. *Communications of the ACM*, 34(10):110–120, 1991.

[56] Kairan Sun, Zhengzi Xu, Chengwei Liu, Kaixuan Li, and Yang Liu. Demystifying the composition and code reuse in solidity smart contracts. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 796–807, 2023.

[57] Ali Sunyaev and Ali Sunyaev. Distributed ledger technology. *Internet computing: Principles of distributed systems and emerging internet-based technologies*, pages 265–299, 2020.

[58] Ali Sunyaev and Ali Sunyaev. Introduction to internet computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, pages 1–24, 2020.

[59] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[60] Adam Toczylowski, Lisa Bacon, Gerald Chew, John Haggerty, and Timur Yontar. Real assets. *Available at SSRN 3154635*, 2018.

[61] Palina Tolmach. Securing smart contracts with formal verification and automated program repair. 2023.

[62] Cryptocurrency Trading, Miguel Ottina, Peter Johannes Steffensen, and Jesper Kristensen. Automated market makers.

[63] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 3–42. Springer, 2012.

[64] Hal R Varian. The arbitrage principle in financial economics. *Journal of Economic Perspectives*, 1(2):55–72, 1987.

[65] Shih-Hung Wang, Chia-Chien Wu, Yu-Chuan Liang, Li-Hsun Hsieh, and Hsu-Chun Hsiao. Promutator: Detecting vulnerable price oracles in defi by mutated transactions. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 380–385. IEEE, 2021.

[66] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[67] Sam Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William Knottenbelt. Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46, 2022.

[68] Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 55(11):1–50, 2023.

[69] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE transactions on software engineering*, 47(10):2084–2106, 2019.