TULIP: A DECENTRALIZED PROTOCOL FOR TOKEN EXCHANGE

GIACINTO PAOLO SAGGESE* AND PAUL SMITH*

Abstract. Limit orders are a natural and powerful way to interact with financial markets. Creating decentralized exchanges around limit orders poses unique challenges, and the dominant decentralized alternatives lack this interface. We propose "Tulip", a protocol for decentralized exchange built around limit orders, at whose heart is a novel auction mechanism for price discovery, matching and clearing. Besides providing a natural trading interface, Tulip promotes welfare maximization, pools liquidity across tokens and time, and has the features of high trading efficiency, complete transparency, no impermanent loss, no need for intermediary custody, no rent extraction from high-frequency traders, censorship resistance, and ease of use. These features have immediate application in improving the efficiency of decentralized token exchange.

Contents

1. Introduction

Tulip is a new financial primitive that enables decentralized token exchange using limit orders. It brings together the advantages of trading efficiency and ease-of-use of centralized exchanges with the transparency and self-custody of decentralized approaches, without the risk of impermanent loss, contrary to automatic market maker-based approaches.

Some of the innovative features of Tulip are:

(1) Support for limit orders: Tulip uses limit orders as a way to express interest in trading. This is the approach that is most natural and in fact used by virtually the entirety of traditional finance, but surprisingly ignored by many decentralized primitives
(2) Welfare maximization: Tulip maximizes the aggregated welfare by allowing the relaxation of the constraint of a single clearing price, which has been the standard assumption in traditional finance and inherited by decentralized finance

Date: May 22, 2023.

* The authors contributed equally to this work and are listed alphabetically.

With contributions from Danil Iachmenev, Tamara Jordania Samarth KaPatel, Grigorii Pomazkin, Juraj Smeriga, Daniil Tikhomirov, Nina Trubacheva, and Vladimir Yakovenko,

(3) Liquidity pooling across tokens: Tulip participants are not artificially segregated in separate liquidity pools, but are aggregated in a single optimization problem that handles in a natural way both multi-coin and reverse token exchange
(4) Liquidity pooling across time: Tulip participants provide liquidity only when they wish to trade, and only in the amount that they wish to trade, instead of artificially separating traders as liquidity providers and liquidity takers as is done in automated market makers protocols
(5) High trading efficiency: Tulip uses discrete and frequent auctions for price discovery and efficiently matching supply and demand, allowing trading tokens peer-to-peer while minimizing and sharing spread costs
(6) Complete transparency: Tulip collects on-chain all information for maximum transparency and allows the independent verification of the correctness of any computation moved off-chain for minimizing execution costs due to current limitations of blockchain technology
(7) No impermanent loss: liquidity providers are not exposed to the risk of impermanent loss (a form of unrealized loss that becomes realized upon withdrawing liquidity), unlike the case with other decentralized exchange protocols based on automated market makers
(8) No intermediary custody: exchanged tokens always remain under the control of their respective owners, in contrast to centralized exchanges
(9) No rent extraction from high-frequency traders: in Tulip speed alone does not confer an advantage to traders, as the discrete and frequent auctions prevent predatory tactics such as front-running, limit order scalping, and spoofing, which are known issues seen with continuous limit order books
(10) Censorship resistance and ease-of-use: the interaction between buyers and sellers occurs through smart contracts and a website front-end

1.1. **Token swap as a financial primitive.** Swapping tokens is one of the most important and widely used primitives in decentralized finance. In March 2023, the average daily trading volume in cryptocurrencies was over $34 Billion USD, after reaching a peak of $516 Billion USD per day in May 20, 2021[1].

1.1.1. *Token swap on decentralized exchanges.* Trading volume on decentralized exchanges has been increasing over time and now is approaching 20% of the total traded cryptocurrency volume[2]. As of March 2023, Uniswap has transacted more than $1.4 Trillion USD and performed more than 139 million trades[3]. The move from centralized to decentralized exchanges has been accelerating due to recent scandals involving centralized exchanges (e.g., FTX bankruptcy), regulatory crackdown, and thanks to end-users' increased embrace of the principles of self-custody and decentralization. We believe that this trend towards larger trading volumes will only accelerate in the future.

1.2. **Limit orders as an interface to token swap.** A limit order expresses a commitment to buy or to sell an asset, up to a certain quantity, at a price that is as good as or better than the limit price. As supply and demand are represented by quantity available at a given price, limit orders provide a way for buyers and sellers to participate in price discovery and exchange.

We adopt the perspective that limit orders may be thought of as an interface to market participation. That is, limit orders provide participants with a means to engage with markets. We further promote this interface as one that is natural, flexible, and elegant. It is natural in that it expresses supply or demand directly in terms of quantity and price, with a clearly defined maximum possible

[1]https://www.statista.com/statistics/1272903/cryptocurrency-trade-volume
[2]https://www.theblock.co/data/decentralized-finance/dex-non-custodial/dex-to-cex-spot-trade-volume
[3]https://uniswap.org/

exposure in a commitment to trade. The interface is flexible because it allows participants to construct personalized supply and demand curves by combining multiple limit orders. The elegance follows from the simplicity and ease-of-use of the interface.

1.2.1. *Limit orders in traditional finance and DeFi.* As a consequence of these benefits, limit orders and their aggregation into limit order books (LOBs) are ubiquitous in traditional financial markets and dominate cryptocurrency trading on centralized exchanges (CEXs). Interestingly, however, limit orders currently do not play as prominent a role in the realm of decentralized finance. Instead, alternative approaches that rely upon automated market makers (AMMs) have taken center stage, and these approaches offer a different set of advantages and disadvantages.

1.3. **How exchanges create value.** To better understand the comparative advantages and disadvantages of different approaches to token exchange (or swap), we step back and consider the purpose and functions of exchanges.

1.3.1. *Market participants.* Exchanges create value by bringing together different types of participants, such as investors, traders, hedgers, brokers, arbitrageurs, and market makers. Market participants have different goals (e.g., hedging, investing, speculating), horizons (from low-latency trading to long-term investment over multiple years), risk tolerance, liquidity preferences, and beliefs about security values. The opportunity to trade arises from these differences.

1.3.2. *Roles of market exchanges.* Exchanges have several roles:

- Provide a common meeting ground for market participants to facilitate matching supply and demand
- Establish the rules of the trading process and institutional roles, so that the trading process is structured, monitored, and standardized
- Provide a certain amount of oversight by monitoring and certifying financial statements and governance procedures
- Generate market data for market participants, such as trades and quote changes

Exchanges are compensated for this value creation by collecting transaction fees as a small percent of traded volume.

Typically, an exchange requires two components:

(1) Trade matching: participants find a counterparty agreeing on quantity and price of the assets to swap (this includes price discovery)
(2) Trade settlement (clearing): the assets are actually swapped after finding matching counterparties

1.3.3. *Exchanges in decentralized finance.* In decentralized finance, a token swap can be accomplished in different ways, e.g., using

- a centralized exchange (CEX) (e.g., Coinbase, Binance, OkX)
- a decentralized exchange (DEX) (e.g., Uniswap, SushiSwap)

We consider CEXs as a temporary bridge between traditional finance and the new vision of decentralized finance and we do not consider them a viable long-term solution to the problems that DeFi is poised to address. Centralized exchanges are usually organized around different versions of a central limit order book (CLOB).

Decentralized exchanges can be organized as:

- off-chain order books (e.g., 1inch)
- on-chain order books (e.g., SwapSwap, KyberSwap)
- automated market makers (AMM) (e.g., Uniswap)

Off-chain solutions can run into custodial and censorship issues that stand in direct conflict with the ethos of decentralization and self-determination. Unfortunately, limitations of current blockchain technology restrict the amount of computation that can be performed on-chain, although these limits are continously being removed by advancements in research and blockchain technology (e.g., layer 2 chains, optimistic and zero-knowledge rollups). Thus we consider solutions based on self-custody that do not require trusting a third party (or that at least allow one to verify its fairness) in line with the principles of decentralization.

Off-chain order books are an hybrid version of CEX and DEX where the price discovery and trade matching happens off-chain, while the trade settlement is performed on-chain.

On-chain order book matching has the advantages of being custodial, since users are completely in charge of their funds. A major disadvantage of using on-chain limit order books in decentralized exchange is cost of computation. Continual submission and cancelation of orders incurs costs, as does the ongoing process of matching supply and demand.

1.4. **The advantages of Tulip.** Tulip retains the key advantages of on-chain decentralized exchange while overcoming the issues of cost. Additionally, it includes an implementation that preserves the familiar LOB interface but deepens the pool of available liquidity in matching supply and demand.

Behind the interface, we propose two smart contracts[4] for ERC-20[5] token exchange that match supply and demand differently depending upon whether there exists an external reference price (TulipCross) or whether the contract also performs the role of price discovery (TulipSwap).

1.4.1. *TulipCross.* TulipCross is a decentralized liquidity pool that crosses buy and sell orders with respect to an external reference price, i.e., a price oracle. This allows traders to interact directly with each other and share price improvement with respect to exchanges, e.g., by trading at bid-ask midpoint.

1.4.2. *TulipSwap.* TulipSwap performs price discovery by matching supply and demand at regular time intervals. In the simplest case, it resembles a Walrasian-style auction [21]. In cases where liquidity is pooled more broadly, the auction clearing mechanism satisfies a collection of inequalities.

1.4.3. *Comparison with alternative solutions.* A comparison between TulipCross and TulipSwap and other solutions for token swapping, namely limit-order books (e.g., centralized exchanges such as Binance) and automatic market-makers (e.g., Uniswap) is summarized below and discussed in detail in Section 8.

	LOB	*AMM*	*TulipSwap*	*TulipCross*
Support for limit orders	+	-	+	+
Welfare maximization	o	-	+	-
Liquidity pooling across tokens	-	o	+	+
Liquidity pooling across time	+	-	+	+
Trading efficiency	o	-	+	o
Transparency	-	+	+	+
No impermanent loss	-	-	+	+
Low custodial risk	-	+	+	+
No rent extraction	-	-	+	o
Censorship resistance and ease of use	-	+	+	-

[4]https://ethereum.org/en/developers/docs/smart-contracts/

[5]https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

2. Matching liquidity

Suppose there is a group of `ETH` holders and a group of `wBTC` holders who wish to swap tokens at a competitive rate. How should the liquidity be pooled? At first glance, it appears that there should be a single pool of liquidity. The wrinkle manifests in determining how to express a desire to trade. A simple expression of a desire to trade is a limit order representing a commitment to trade up to q in quantity of a token at a token exchange rate up to p. But what if some parties express quantity in terms of `ETH` and some in terms of `wBTC`, and some parties express limit prices in terms of `ETH` per `wBTC`, while others express limit prices in terms of `wBTC` per `ETH`? Under these conditions, setting a single clearing price and determining fill prioritization rules raises several questions. We shall consider and address these in the sequel, but to begin, we discuss a more constrained setting, common in practice.

2.1. **Conventions in foreign exchange markets.** To simplify, we follow the practice of foreign exchange (FX) markets, which break the symmetric, interchangeable roles of two currencies by specifying two non-interchangeable roles:

(1) a *base* currency (for quantity)
(2) a *quote* currency (for price)

The roles are expressed by writing *base currency/quote currency*, e.g., "EUR/USD".

A limit order then consists of a quantity expressed in the units of the base currency and a price expressed in terms of the quote currency (per unit or suitable multiple of base currency). See, for example, [8] (e.g., footnotes on page 25) for usage of this terminology. See [9] for how this works in practice for Euro FX contracts, where contract units are denominated in Euros and price is quoted in U.S. dollars and cents per Euro increment.

By breaking the symmetry, one may run a standard limit order book, hold a classical Walrasian-style auction, and handle size quantization effects by using one of a variety of priority rules based on quantity, price, and time. See [4] for discussion around priority rule variations and their effects on market quality outcomes.

An effect of breaking the symmetry in this way is that either liquidity for a pair of currencies may be expressed in only one of the two forms, or liquidity for a pair of currencies is split across two separate contracts (with roles of base and quote token reversed), each with its own limit order book. Using our example, this would mean treating `wBTC/ETH` and `ETH/wBTC` as separate token pairs, even though the union of the underlying parties and sources of liquidity are typically the same.

2.2. **Multi-coin exchange.** It seems plausible that by retaining the symmetry in the two-token case and instead treating a pair of tokens as a single source of liquidity, one may devise a more efficient exchange mechanism.

This minor expansion, however, suggests widening even further the universe of eligible swaps. For example, if there are three tokens available for exchange, one could contemplate many-for-one, one-for-many, or many-for-many token swaps, all to be carried out in an atomic fashion, and perhaps with complex constraints. A toy case along these lines one may consider is as follows:

Problem 2.2.1 (Triangular liquidity)**.** Suppose there are the following three parties:

- Party A, who holds `wBTC` and wants `ETH`
- Party B, who holds `ETH` and wants `DAI`
- Party C, who holds `DAI` and wants `wBTC`

In the case above no party can trade with another single party, and yet there is a way to collectivily swap tokens that satisfies all parties. How should an auction be structured to "best" facilitate exchange?

While there may be other use cases and even demand for such auctions, unfortunately, the problem in general is NP-complete (e.g., [22]), which is a significant limiting factor in terms of feasibility and auction expense.

In the sequel, we propose practical exchange mechanisms valid for any collection of standard limit orders (all base/quote token limit orders may be pooled and cleared in a single optimization).

3. Limit orders

In the previous section, we discussed the notions of base currency and quote currency from foreign exchange. Here we extend them to tokens.

3.1. **Base/quote tokens.** A token (ordered) pair consists of a *base token* and a *quote token*. The shorthand notation for expressing the pair is *base token/quote token* (e.g., "`wBTC/ETH`"), and as such indicates the respective roles of the tokens.

By convention, *quantity* to exchange is expressed in terms of the base token, and *price* is a token exchange rate expressed in terms of the quote token per unit of base token (or multiple thereof).

A party who places a "buy" order must have the quote token (`ETH`) in custody, i.e., in its wallet. A party who places a "sell" order must have the base token (`wBTC`) in custody.

Example 3.1.1 (base/quote tokens). In the pair `wBTC/ETH`, `wBTC` is the base token and `ETH` is the quote token. Interest to trade is expressed in terms of orders with `wBTC` as base token and `ETH` as quote token, as follows:

- A "buy" order represents a commitment to purchase the base token `wBTC` and pay with the quote token `ETH`
- A "sell" order represents a commitment to sell the base token `wBTC` and receive the quote token `ETH`
- Quantity q is in terms of the base token ("buy/sell a certain amount of `wBTC`")
- Limit price is in terms of the quote token ("buy/sell `wBTC` with a limit price of p `ETH` per `wBTC`")

3.2. **Price properties.** If prices of base and quote tokens are expressed in terms of a common currency (e.g., USD) then it holds that:

$$p_{quote\_per\_base} = \frac{p_{quote}}{p_{base}},$$

where the numeraire plays the role of a quote token for both p_{quote} and p_{base} and is implicit.

Of course, it holds that

$$p_{quote\_per\_base} = \frac{1}{p_{base\_per\_quote}}.$$

Note that in the following we use both notations of price of a token relative to another token, and price of a token with respect to a common numeraire, depending on which notation is simpler. The two representations are equivalent aside from a multiplicative constant, and it is always possible to convert one into the other.

Centralized exchanges typically present token prices with a USD-based stablecoin as the quote token, so that, in effect, the US dollar is the numeraire.

3.3. **Order attributes and notation.** We introduce the following tuple notation for a general limit order (valid for both TulipCross and TulipSwap).

Definition 3.3.1 (Limit order)**.** A TulipCross or TulipSwap limit order is represented by a tuple of the form

(`timestamp`,
`action`,
`quantity`,
`base_token`,
`limit_price`,
`quote_token`,
`deposit_address`)

The quantities are arranged to make the order simple to read in natural language: "At timestamp `timestamp` create an order to `action` up to a number `quantity` `base_token` tokens for a limit price of `limit_price` with respect to the token `quote_token` and deposit the resulting tokens at `deposit_address`."

We have previously discussed the roles of `action`, `quantity`, `base_token`, `limit_price`, and `quote_token`. The roles of `timestamp` include determining eligibility in a swap and can extend to influencing order fill priority. The `deposit_address` attribute mirrors standard functionality available in traditional finance, e.g., in purchasing T-Bills on TreasuryDirect[6]. This feature facilitates account management through the use of multiple wallets. For example, a miner may have a supply of `wBTC` collected and held in one wallet which it would like to systematically diversify into `ETH` and perhaps other tokens. In specifying a deposit address, it may use a separate wallet to collect incoming `ETH`.

Example 3.3.1 (Limit order notation)*.* The order

(`1678660406`, `buy`, `3.2`, `ETH`, `4.0`, `wBTC`, `0xdeadc0de`)

corresponds to the natural language description: "At timestamp Mon Mar 13 2023 02:33:25 GMT+0000, the user commits to buy up to 3.2 units of `ETH` in exchange for `wBTC` up to a `limit_price` of 4.0 `wBTC` per `ETH` with proceeds deposited at `0xdeadc0de`".

3.3.1. *Order shorthand notation.* In the sequel, when clear from the context we may:

- omit `timestamp` and `deposit_address` when not relevant to the discussion
- omit the (infinite) `limit_price` for market orders

3.3.2. *Extracting attributes from an order.* A limit order can also be represented as a tuple in the following way

$$o = (a, q, \pi, p, \tau)$$

where

- a is the desired action (`buy` or `sell`)
- π is the base token
- τ is the quote token
- q is the maximum quantity desired in the exchange of the base token π
- p is the limit price in terms of quote token τ per base token π

Additionally, for each o, let a_π and a_τ belong to $\{-1, +1\}$, with sign determined by order o's action a in the following way: if a (the action) of o is to buy, then set $a_\pi = -1$ (the buy order is taking liquidity of the base token), and if it is to sell, then set $a_\pi = 1$. We always have $a_\tau = -a_\pi$, and if the action of a is to sell rather than to buy, then its signs are reversed with respect to the buy action.

[6] https://www.treasurydirect.gov

This compact notation is more convenient for expressing and analyzing constraints imposed by limit orders.

3.4. **Building a supply/demand curve with limit orders.** A market participant may express a personalized supply or demand curve with limit orders.

Suppose price p may vary in discrete price increments (the case in practice), and let $Q_+(p)$ denote the (total) quantity market participant Alice desires at price p (or better). We make the mild (and rational) assumption that $Q_+(p)$ is monotonically decreasing in p: as price p goes up, Alice desires to buy less in total quantity.

To translate Alice's demand into a set of limit orders, first assume that there is a maximum price that Alice is willing to pay, p_M (if no such upper limit exists, we may first assume that Alice executes market orders until such a price exists). Assume also that there is a maximum quantity Q_M that Alice demands, and that price is always positive. Then, there exists a partitioning of prices $(p_j, p_{j+1}]$ and positive (cumulative) quantities Q_j such that

$$Q_+(p) = \sum_{j=0}^{n} Q_j \cdot \mathbb{1}_{(p_j, p_{j+1}]}(p),$$

where $\mathbb{1}$ denotes the indicator function. We allow $p_{j+1} = p_j$. This representation simply expresses the fact that $Q_+(p)$ only changes at discrete price points (we have assumed price is discrete).

To convert $Q_+(p)$ into a set of limit orders, we begin with the highest price with demand, p_{n+1}, and create the buy limit order o_n with limit price p_{n+1} and quantity Q_n. Next, we create buy limit order o_{n-1} with limit price p_n and quantity $Q_{n-1} - Q_n$. Generically, limit order o_k is a buy limit order with limit price p_{k+1}:

$$p(o_k) = p_{k+1}$$

and quantity $Q_k - Q_{k+1}$:

$$q(o_k) = Q_k - Q_{k+1}$$

Because the quantities Q_j represent cumulative quantity at price p_{j+1} or better, and are monotonically decreasing, each $q(o_k) > 0$.

Note that the case for expressing a supply curve is analogous, and sell limit orders are used instead. The preceding discussion has a continuous analogue, though in practice minimal tick increments mean that the setting is discrete. The key assumption, almost always satisfied in practice, is that cumulative supply and demand are monotonic in price.

3.5. **Order clearing quantity and exchange rate.** In Tulip, orders are collected from users during a finite period of time, after which tokens are redistributed among users according to their limit orders.

Effective prices for the tokens are determined based on the available limit orders, constraints defining different swap problems, and an optimization criterion designed to maximize the welfare of the participants in the swap. At the same time, orders are matched in (generically) many-to-many relationships.

Swaps between base/quote token pairs can occur

- at a single exchange rate, i.e., the clearing (or equilibrium) price is the same for all executed orders; or
- at different exchange rates for different orders

A limit order's price constraint may not be violated in an execution for that order.

In constrast, quantity exchanged is always specific to each order. Each order's constraint on quantity exchanged cannot be violated, and that order may participate in exchange only if its effective price respects its limit order price.

3.5.1. *Attributes of an executed order.* When an order o is executed, quantity exchanged (of both base and quote tokens) is observable, and, in the event of an execution, an effective price for that order is observable from the amount of base and quote token exchanged. Moreover, in some exchange mechanisms, there is a unique global clearing price for any pair of tokens exchanged, and this also becomes observable.

We qualify these observed values with a star superscript on the corresponding limit order value. Recalling that an order may be represented as the tuple

$$o = (a, q, \pi, p, \tau),$$

we let

- q^*_π be the quantity of the base token π exchanged in the execution of order o
- q^*_τ be the quantity of the quote token τ exchanged in the execution of order o
- p^* be the effective price for the order, given by q^*_τ / q^*_π, in the event that q^*_π is positive (otherwise it is undefined)
- $p^*_{quote\_per\_base}$ denote the unique exchange rate between the quote and base tokens for all the swaps (if it exists)
- $p^*_{quote\_per\_base}$ denotes the unique exchange rate between the quote and base tokens for all the swaps (if it exists)

By convention, we require that

$$q^*_\pi \geq 0, \quad q^*_\tau \geq 0.$$

3.6. **Execution of orders.** Next we introduce some examples of market order and limit order behavior when a clearing exchange rate has been set. In particular, consider a swap for the tokens `ETH`, `wBTC` and assume that the exchange rate between `ETH` and `wBTC` is fixed at 0.2 (i.e., 0.2 `wBTC` can be exchanged for 1 `ETH` and vice versa).

Example 3.6.1 (Market order notation and clearing). The following market orders omit `timestamp`, `limit_price`, and `deposit_address` in favor of emphasizing the amounts of token exchanged:

- An order `(buy, 1.0, ETH, wBTC)` means buying 1 `ETH` in exchange for 0.2 `wBTC`
- An order `(sell, 1.0, ETH, wBTC)` means selling 1 `ETH` and receiving 0.2 `wBTC`
- An order `(buy, 1.0, wBTC, ETH)` means buying 1 `wBTC`, paying with 5 `ETH`
- An order `(sell, 1.0, wBTC, ETH)` means selling 1 `wBTC` in return for 5 `ETH`

Next we consider the behavior of limit orders under the same prevailing exchange rate and we assume that there is sufficient supply of tokens to fully fill the orders.

Example 3.6.2 (Executable buy limit order). A limit order

`(buy, 1.0, ETH, 0.5, wBTC)`

means "buy up to 1 `ETH` in exchange for `wBTC` at a rate up to 0.5 `wBTC` per `ETH`." In this case, since the price of one `ETH` is equal to 0.2 `wBTC`, the order can be executed at the prevailing market rate.

Example 3.6.3 (Non-executable buy limit order). On the other hand, a limit order

`(buy, 1.0, ETH, 0.1, wBTC)`

requires that the rate of `wBTC` per `ETH` be lower than the current market rate, and so the limit price prevents a token swap from being carried out.

Example 3.6.4 (Non-executable sell limit order). A limit order

`(sell, 1.0, ETH, 0.5, wBTC)`

means "sell up to 1 unit of ETH in exchange for wBTC at a rate down to 0.5 wBTC per ETH." Since the current rate of wBTC per ETH is 0.2, which is below the limit price of 0.5, the order cannot be executed.

4. Constraints

Throughout this section, let $\{o_i\}_{i=1}^n$ denote a collection of n limit orders, each of the form

$$o_i = (a_i, q_i, \pi_i, p_i, \tau_i),$$

where the quote token π_i is implicit in the quantity q_i and the quote token τ_i and base token π_i are implicit in the limit price p_i and in the action a_i.

Recall that for each order o_i, $q^*_{i,\pi}$ denotes the realized amount of base token exchanged, and $q^*_{i,\tau}$ denotes the realized amount of quote token exchanged. Both values are always well defined and nonnegative.

4.1. **Limit order constraints always respected.** Each limit order constraint on quantity exchanged is straightforward:

$$q^*_{i,\pi} \leq q_i. \tag{1}$$

The limit order price constraint inequality direction depends upon whether the action is to buy or to sell. The effective transaction price (in the event that positive tokens are exchanged) is represented by $q^*_{i,\tau}/q^*_{i,\pi}$. If the action is to buy (base token), then p_i represents an upper bound, and the opposite is true if the action is to sell:

$$\begin{cases} q^*_{i,\tau} \leq q^*_{i,\pi} p_i, & a_{i,\tau} = 1 \\ q^*_{i,\tau} \geq q^*_{i,\pi} p_i, & a_{i,\tau} = -1 \end{cases} \tag{2}$$

Here we have multiplied through by $q^*_{i,\pi}$, with the result being that the inequality also holds in the event that $q^*_{i,\pi}$ or $q^*_{i,\tau}$ is zero.

4.2. **Supply and demand always match.** All of the exchange mechanisms that we consider must match quantity supplied with quantity consumed in a swap. That is, the exchange neither injects nor removes token from the swap. To formally express this, we introduce the following notation.

Let T denote the intersection of the union of all base tokens and the union of all quote tokens:

$$T := \left(\bigcup_{i=1}^n \pi_i\right) \bigcap \left(\bigcup_{i=1}^n \tau_i\right)$$

The set of tokens T is the set of tokens eligible for swapping.

For each token $u \in T$, define the indicator function $\mathcal{T} : T \to \{0, 1\}$ via

$$\mathcal{T}_u(v) = \begin{cases} 1 \text{ if } v = u \\ 0 \text{ if } v \neq u \end{cases}$$

Then, for each $u \in T$, the following conservation law must hold:

$$\sum_i \mathcal{T}_u(\pi_i) \cdot a_{i,\pi} q^*_{i,\pi} + \mathcal{T}_u(\tau_i) \cdot a_{i,\tau} q^*_{i,\tau} = 0 \tag{3}$$

Note that this translates into one equality per token participating in the swap. These equalities express the notion that each filled order must have, across the collection of orders, suitable counterparties. In other words, for each token, the total amount bought must equal the total amount sold.

4.3. **Unique clearing price per token pair.** In the TulipCross case, a single exchange rate per exchanged token pair prevails, as a reference price serves this function. One may make the case that this is a desirable property for token swap more generally. That is, for participants trading the same token pair, no participant gets a price that is any better or any worse than what any other participant gets.

Suppose that the effective exchange rate for an order involving tokens $\nu \in T$ and $\eta \in T$ is the same for all filled orders. In other words, the clearing exchange rate is unique for all participants. Then, for all i, j whose respective orders o_i, o_j both involve tokens ν, η, we have

$$q^*_{i,\pi} q^*_{j,\tau} - q^*_{i,\tau} q^*_{j,\pi} = 0$$

when the tokens ν and η play identical base/quote roles in o_i, o_j, and

$$q^*_{i,\pi} q^*_{j,\pi} - q^*_{i,\tau} q^*_{j,\tau} = 0$$

if they play opposite roles.

We rewrite this as

$$\begin{cases} q^*_{i,\pi} q^*_{j,\tau} - q^*_{i,\tau} q^*_{j,\pi} = 0, & \forall i, j \in \{1, \ldots, n\} : \pi_i = \pi_j \wedge \tau_i = \tau_j \\ q^*_{i,\pi} q^*_{j,\pi} - q^*_{i,\tau} q^*_{j,\tau} = 0, & \forall i, j \in \{1, \ldots, n\} : \pi_i = \tau_j \wedge \tau_i = \pi_j \end{cases} \tag{4}$$

The unique clearing price constraint prevents the problem from falling in the domain of linear programming (to be discussed in the sequel).

4.4. **No arbitrage constraint.** In TulipCross, multi-currency arbitrages may or may not exist in the reference prices. In practice, one would expect such opportunities in the reference prices to be fleeting or small, since otherwise arbitrageurs would exploit them until market pressures eliminated the opportunities. Note, though, that it is not the exchange mechanism itself that minimizes arbitrage opportunities; rather, it is the active participation of arbitrageurs.

In the case of multi-token TulipSwap, and in the case where we impose unique clearing prices (per token pair), we have the flexibility to impose the additional constraint of no-arbitrage in the realized prices. One the one hand, this property is appealing in terms of fairness. On the other hand, any such arbitrages are not exploitable in practice (the swap is atomic), and in any case, the additional constraints are nonlinear in terms of how they constrain the problem and nonlinear in terms of their growth in the number of distinct tokens. We present how such constraints may be expressed, if desired.

Let Γ denote the directed graph whose vertices V are given by the set of elements T, and whose directed edges E are given by

$$E = \bigcup_{\substack{i \in \{1,\ldots,n\} \\ \pi_i, \tau_i \in T}} ((\pi_i, \tau_i) \cup (\tau_i, \pi_i))$$

Each edge $(b, a) \in E$ we give the weight $w_{a,b}$, corresponding to a log exchange rate. Note that $w_{a,b}$ and $w_{b,a}$ are distinct.

Let C denote the collection of all simple cycles of Γ. For an simple cycle $\gamma \in C$, consecutively enumerate the vertices (with starting place arbitrary) by c_j for $j = 1, \ldots, |\gamma|$. Let $c_0 := c_{|\gamma|}$ for notational convenience. Then the *no arbitrage* constraints state that, for each $\gamma \in C$, we have

$$\sum_{j=0}^{|\gamma|} w_{j,j+1} = 0$$

In the case of a two-token cycle, this constraint reduces to ensuring that there is a unique exchange rate between the pair of tokens (changing the roles of the base and quote tokens in limit orders

inverts the exchange rate). For an efficient algorithm for finding all elementary cycles, see [12] as well as [14] for a comparison of various algorithms.

4.5. **Maximization criteria in a token swap.** In a Walrasian auction, a single clearing price is chosen so that, at that price, aggregate supply meets aggregate demand (with priority rules used in the discrete setting to handle discretization effects). This may be framed as a max-min problem (see [5][§1.1.1]).

We propose maximizing the total quantity of token swapped, which is a simple linear constraint. That is, the objective to maximize is

$$\sum_{i=1}^{n} q^*_{i,\pi} + q^*_{i,\tau} \tag{5}$$

4.5.1. *Invariance under token redenomination.* Note that any feasible solution that satisfies the constraints is invariant under the change of scale of a token, e.g., if the unit of measurement of quantity of a token changes, and if the limit orders are updated accordingly with the unit change, then any solution to the original problem remains a solution to the rescaled problem.

What remains to show is that the optimal solution to the original problem transforms into the optimal solution to the rescaled problem (even though the optimal values of the objective (5) are generally different). For this, we may take a feasible solution to the original problem and map it to its corresponding feasible solution in the rescaled problem. The only part of the objective function that changes is the contribution of the rescaled token, which is rescaled linearly. The property of being a local maximum is preserved by this transformation (which can be seen by considering partial derivatives), and since any local maximum is a global maximum (due to convexity), the transformed solution is the optimal solution to the transformed problem.

4.5.2. *Is a single clearing price needed?* Maximizing token swapped subject to the basic limit order and token conservation constraints does not necessarily lead to unique clearing prices for each token pair. In fact, adding the additional constraints associated with a unique clearing price would reduce the total volume exchanged. We argue that by not imposing a single clearing price, more of each participant's supply or demand curve may be satisfied. As an added benefit, the swap problem to solve is a linear program rather than an optimization problem with nonlinear constraints.

4.6. **Alternative expression of limit order constraints.** Limit orders alternatively may be translated into inequalities involving quantity of tokens exchanged and token exchange rates. This formulation may be more intuitive in cases where a unique token clearing price should prevail.

4.6.1. *Inequalities for buy order.* An order of the form $o_i =$ `(buy, q, A, p, B)` means "buy q units of token `A` in exchange for token `B` with a limit price up to p `B` per unit of `A`", and it corresponds to the following constraint on realized quantity exchanged and clearing exchange rate:

$$(p^*_{\mathtt{B\_per\_A}} \leq p(o_i)) \wedge (0 \leq q^*_{\mathtt{A}}(o_i) \leq q(o_i)) \vee (q^*_{\mathtt{A}}(o_i) = 0)$$

The constraint on the quantity exchanged is conditioned on the corresponding price's satisfying the desired limit price constraint: if the desired limit price constraint is not met, the swap cannot be carried out and the exchanged quantity is 0.

Example 4.6.1 (Inequalities for buy order). An order of the form $o_1 =$ `(buy, 2, A, 3, B)` means "buy 2 units of token `A` in exchange for token `B` with a limit price up to 3 `B` per unit of `A`", and it corresponds to the following constraint on realized quantity exchanged and clearing exchange rate:

$$(p^*_{\mathtt{B\_per\_A}} \leq 3) \wedge (0 \leq q^*_{\mathtt{A}}(o_1) \leq 2) \vee (q^*_{\mathtt{A}}(o_1) = 0)$$

Example 4.6.2 (Inequality for buy market order). A market order (i.e., without a limit price) has no constraint on exchange rate and thus is reduced to

$$0 \leq q^*_{\mathtt{A}}(o_1) \leq q(o_i)$$

4.6.2. *Inequalities for sell order.* An order of the form $o_i =$ `(sell, q, A, p, B)` means "sell q units of token `A` in exchange for token `B` with a limit price of at least p `B` per unit of `A`", and it corresponds to the following constraint on realized quantity exchanged and clearing exchange rate:

$$(p^*_{\mathtt{B\_per\_A}} \geq p(o_i) \wedge (0 \leq q^*_{\mathtt{A}}(o_i) \leq q(o_i)) \vee (q^*_{\mathtt{A}}(o_i) = 0)$$

Example 4.6.3 (Inequalities for sell order). In the same way, an order like $o_2 =$ `(sell, 3, A, 2, B)` means "sell 3 units of token `A` in exchange for token `B` with a limit price of at least 2 `B` per unit of `A`", which corresponds to

$$(p^*_{\mathtt{B\_per\_A}}(o_2) \geq 2) \wedge (0 \leq q^*_{\mathtt{A}}(o_2) \leq 2) \vee (q^*_{\mathtt{A}}(o_2) = 0)$$

4.7. **Order normalization.** When the exchange rate between two tokens is known, then it is possible to convert buy/sell orders for both `A` and `B` tokens into buy/sell orders where all orders have token `B` as the base token, according to the transformation below.

Let

$$o_1 = (\mathtt{buy}, q, \mathtt{A}, p_{\mathtt{B\_per\_A}}, \mathtt{B}).$$

Suppose that $p^*_{\mathtt{B\_per\_A}}$ is fixed and known. Then the order

$$o_2 = (\mathtt{sell}, q', \mathtt{B}, 1/p_{\mathtt{B\_per\_A}}, \mathtt{A})$$

may be handled in order crossing in the same way that o_1 may be provided that

$$q' = q \cdot p^*_{\mathtt{B\_per\_A}}$$

5. Formulation and solution of TulipSwap and TulipCross problems

The general problem of determining the token equilibrium prices and the allocation among the swap participants can be formulated in terms of the inequalities on quantities and prices corresponding to the orders and on the need that the total quantity exchanged of each token be preserved across the swaps, i.e., the quantity bought for each token is equal to the quantity sold.

5.1. Problem formulations.

Problem 5.1.1 (TulipSwap token exchange)**.** Let $\{o_i\}_{i=1}^n$ be a collection of limit orders, with associated constraints given by (1), (2), and (3). The TulipSwap problem is to find nonnegative quantities $q^*_{i,\pi}, q^*_{i,\tau}$, $i = 1, \ldots, n$, that satisfy the constraints (1), (2), and (3), such that the objective function (5) is maximized.

Problem 5.1.2 (TulipCross token exchange)**.** Let $\{o_i\}_{i=1}^n$ and associated constraints be as in Problem 5.1.1. Let $\{p^*_i\}$ denote a collection of reference prices, such that each p^*_i only depends upon the base token π_i and quote token τ_i of order o_i. Introduce the reference price constraint

$$a_{i,\tau} q^*_{i,\tau} = q^*_{i,\pi} p^*_i, \quad i \in \{1, \ldots, n\} \tag{6}$$

The TulipCross problem is to find nonnegative quantities $q^*_{i,\pi}, q^*_{i,\tau}$, $i = 1, \ldots, n$, that satisfy the constraints (1), (2), (3), and (6) such that the objective function (5) is maximized.

For non-compatible limit orders, the only quantities that satisfy both the limit order inequality (2) and the reference price equality (6) are given by

$$q^*_{i,\tau} = q^*_{i,\pi} = 0$$

Problem 5.1.3 (TulipSwap token exchange with unique clearing prices)**.** Let $\{o_i\}_{i=1}^n$ and associated constraints be as in Problem 5.1.1. The TulipSwap problem with unique clearing prices is to find nonnegative quantities $q^*_{i,\pi}, q^*_{i,\tau}$, $i = 1, \ldots, n$, that satisfy the constraints (1), (2), and (3), (4) such that the objective function (5) is maximized.

5.2. **Problem solutions.**

5.2.1. *TulipSwap as a linear program.* Let x denote the vector with structure

$$x = \left[q^*_{1,\pi}, q^*_{2,\tau}, \ldots, q^*_{n,\pi}, q^*_{n,\tau}\right]^T$$

Next we introduce the following matrices.

Let Q be the matrix whose i-th row is n-dimensional, with a "1" in the column $2i-1$ and zeros elsewhere, and let q denote the vector

$$x = \left[q_1, \ldots, q_n\right]^T$$

Then the limit order quantity constraint (1) is expressed in matrix form via

$$Qx \leq q.$$

Let P be the matrix whose i-th row is n-dimensional with entry $-p_i$ in the $2i-1$ column, $a_{i,\tau}$ in the $2i$ column, and zeros elsewhere. Then the limit order price constraint (2) may be expressed as

$$Px \leq 0.$$

For (3), we introduce a matrix R, which expresses the equality constraint (which could be expressed as two inequality constraints in canonical form) such that it becomes

$$Rx = 0.$$

We leave the definition of R implicit, though this can be formalized as was done for P and Q.

Then Problem 5.1.1 may be expressed as the problem of maximizing

$$\mathbb{1}^T x$$

subject to the inequality constraints

$$\begin{bmatrix} Q & P & -I \end{bmatrix} x \leq \begin{bmatrix} q & 0 & 0 \end{bmatrix}$$

and the equality constraint

$$Rx = 0.$$

Note that this problem falls in domain of linear programming and may be solved via standard simplex or interior point methods.

The problem is polynomial-time in the ideal setting of infinite divisibility. Discretization (and renormalization) move the problem to the domain of integer programming, which is NP-complete. In our setting, where the typical exchanged value is much larger than the discretization grid, obtaining a linear programming solution (i.e., solving the linear programming relaxation) and discretizing after-the-fact may be acceptable in practice.

5.2.2. *TulipCross as a linear program with additional constraints.* We proceed as in the case of TulipSwap, but add the additional equality constraint (6), which may similarly may be expressed in matrix form.

As in the case of TulipSwap, TulipCross is a linear program, and so may be solved efficiently.

5.2.3. *TulipSwap with unique clearing prices.* Problem 5.1.3 differs from Problem 5.1.1 in that constraints (4) must hold. These constraints are quadratic rather than linear in x, which means that Problem 5.1.3 falls outside of the domain of linear programming.

Due to the fact that the constraint matrix is not positive definite and the fact that the matrix expresses a constraint rather than part of an objective function, it does not appear that the problem admits a quadratic programming formulation.

6. TulipCross order matching with two tokens

In this section, we consider a simplified version of Problem 5.1.2. In particular, we only allow two tokens in the pool of liquidity, and we use order normalization to transform orders so that there is a unique base token and unique quote token across all orders.

6.1. **TulipCross reference clearing prices.** TulipCross relies on a *price oracle* for determining the effective token exchange rate in a cross. The price oracle may come from a lit exchange, centralized or decentralized, or even on-chain automated market makers such as Uniswap ([2, §2.2]).

6.2. **Source of reference price.** The case of using an automated market maker as a price oracle is relatively straightforward, provided there is an automated market maker trading the target currency pair with sufficient liquidity.

To use an exchange as a price reference, we must consider some additional steps. First, exchanges typically use a dollar stablecoin (e.g., `USDC`, `USDT`) as the quote currency and all other currencies as base currencies. Our example of `wBTC/ETH` would be considered a cross pair in the world of traditional finance. Because most cross pairs are not traded directly on exchanges, we must derive a clearing price from pairs involving stablecoins.

6.2.1. *Last price as reference price.* One can use as reference price for TulipCross the last executed price on an exchange or AMM.

6.2.2. *Exchange bid-ask midpoint reference price.* Let $\text{bid}_{\text{BTC}}, \text{ask}_{\text{BTC}}$ be stablecoin-denominated top-of-book bid-ask prices for `wBTC` (e.g., expressed in `USDC`), and $\text{bid}_{\text{ETH}}, \text{ask}_{\text{ETH}}$ be the analogous prices for `ETH`. Then, a commitment to buy one `wBTC` and pay `ETH` would clear at a midpoint price of

$$\texttt{wBTC/ETH}_{\text{midpoint}} = \frac{\text{bid}_{\text{BTC}} + \text{ask}_{\text{BTC}}}{\text{bid}_{\text{ETH}} + \text{ask}_{\text{ETH}}}$$

expressed in `wBTC` per `ETH`. Note that if the dollar price of BTH is significantly more than the dollar price of ETH, then this price implies paying many multiples of ETH for BTC (as is the case at the time of this writing).

In general, TulipCross may use the following reference price for a base/quote token pair, where the bids and asks come from a lit exchange and are expressed in terms of stablecoin prices:

$$\frac{\text{bid}_{\text{quote token}} + \text{ask}_{\text{quote token}}}{\text{bid}_{\text{base token}} + \text{ask}_{\text{base token}}} \tag{7}$$

6.2.3. *Averaging reference prices.* When using either an on-chain price oracle or an exchange as a price oracle, it is possible to perform a time or volume weighted average of prices (namely TWAP and VWAP) so as to avoid extreme price variations and to mitigate the risk and effects of any market manipulation attempts. We note, though, that averaging schemes may introduce triangular arbitrages in the reference prices that may not exist point-in-time.

6.3. **Order crossing.** At regular time intervals, order submissions are cut off and reference prices are determined. An element of randomness is used to determine order cutoff times and reference price cutoff times in order to mitigate manipulative behaviors.

An order is eligible for matching if its timestamp is within the cutoff window and if the external reference price does not exceed its limit price.

Except on occasions where eligible buy/sell volume is perfectly matched, not all orders can be fully crossed. There are also discretization effects to consider arising from discrete order sizes and a discrete price grid [5][§3.2.1]. See [4] for a discussion of the trade-offs surrounding different prioritization rule choices. See [18] for the prioritization rules used by one of Morgan Stanley's equity liquidity pools.

6.3.1. *Priority rules.* TulipCross prioritizes fills according to:

- Volume (higher volume comes first in priority)
- Price (higher limit price breaks volume ties)
- Timestamp (earlier timestamp breaks ties in volume and price)

If, in the unlikely scenario that all three of these parameters perfectly agree, a certain priority is not guaranteed.

6.3.2. *Matching algorithm.* The mechanism for matching eligible buy and sell orders consists of two priority queues, one for eligible buy orders and one for eligible sell orders. Priority is determined according to the volume/price/timestamp priority rules introduced above. Top-of-queue orders are compared, and the lesser of the two volumes is fully filled. Once an order is fully filled at the established reference price, it is removed from the priority queue. The procedure continues until one of the two priority queues is empty.

6.3.3. *Computational complexity.* Let n denote the sum (or max) of the number of eligible buy and sell orders. Priority queue construction occurs in $O(n)$ time, each order removal costs $O(\log n)$, and order removal occurs $O(n)$ many times. So, the computational time complexity of the task is $O(n \log n)$. Memory requirements are $O(n)$.

7. Implementation details

7.1. **TulipCross and TulipSwap Architecture.** Tulip addresses cost issues primarily in two ways: (1) discretizing time; and (2) scaling with layer 2 solutions. By addressing these issues, we combine the novel advantages of decentralized exchange with the utility and ease-of-use of the familiar interface of limit orders.

7.2. **Off-chain computation.** Currently TLP offloads some computations to external oracles, due to current computational limitation of blockchains. We do not believe that this is detrimental to the security and decentralization level of TLP as long as these computations are provably correct.

For instance, although the solution of the TulipSwap problem 5.1.1 is polynomial in the number of constraints (or NP-complete in the fully discretized case), verifying that one solution is correct only requires time linear in the number of constraints.

For this reason, TulipSwap stores on-chain the result of the TulipSwap optimization in order to allow independent verification that the off-chain system is not malicious or compromised.

7.3. **Ensuring a timely solution.** TulipSwap avoids the case where solving the optimization problem becomes intractable by performing a swap when a maximum number of orders and/or currency pairs is reached.

8. A comparison of token swapping solutions

In this section, we expand upon the summary table introduced in 1.4.3.

8.1. **TulipSwap.**

- Support for limit orders: Yes
  - TulipSwap supports limit orders natively
- Welfare maximization: Yes
  - The TulipSwap optimization problem allows exchanging tokens with multiple clearing prices to maximize the aggregated welfare of the participants
- Liquidity pooling across tokens: High
  - The TulipSwap optimization problem allows exchanging tokens simultaneously across all participants independently of their base and quote tokens
- Liquidity pooling across time: High
  - In TulipSwap there is no difference between the roles of liquidity providers and liquidity takers, so that takers and makers can trade against each other directly and efficiently
- Trading efficiency: High
  - TulipSwap relies on periodic auctions, and it has been argued in previous literature that for small enough intervals (e.g., seconds) periodic auctions result in the same quality as continuous matching [6]
- Transparency: Yes
  - Price discovery, matching, and clearing are either done completely on-chain or with additional off-chain computations that are independently verifiable. Thus TulipCross guarantees the requirements for a fully transparent solution.
- Impermanent loss: No
  - There is no impermanent loss, since liquidity is not locked inside the contract (contrary to the case for AMMs)
- Custodial risk: No
  - Private keys and funds are always under the control of the users
- Rent extracted by high-frequency traders: Low
  - Discrete and frequent auctions minimize predatory behaviors from HFT behavior [6]
- Censorship resistance: Yes
  - The application runs on-chain and even if the web front-end is attacked or disabled. The off-chain computation can be made robust using similar approaches to distributed oracles, or even ported on-chain if gas prices are not an issue (e.g., using layer 2 solutions).

8.2. **TulipCross.**

- Support for limit orders: Yes
  - TulipCross supports limit orders natively
- Welfare maximization: No
  - The TulipCross optimization problem uses prices determined by a lit exchange and thus there is a single clearing price for all the swaps. This does not allow achieving maximal welfare for the traders involved in the swap.
- Liquidity pooling across tokens: High
  - TulipCross has the same behavior as TulipSwap from this point of view
- Liquidity pooling across time: High
  - TulipCross has the same behavior as TulipSwap from this point of view
- Trading efficiency: Medium
  - The trading efficiency in matching trades is the same as the lit exchange on which price discovery is performed (e.g., a centralized exchange)
- Transparency: Yes
  - TulipCross has the same behavior as TulipSwap from this point of view

- Impermanent loss: No
  - TulipCross has the same behavior as TulipSwap from this point of view
- Custodial risk: No
  - TulipCross has the same behavior as TulipSwap from this point of view
- Rent extracted by high-frequency traders: Medium
  - TulipCross behaves as a dark pool of liquidity and thus it can be susceptible to predatory behaviors, although mitigated by the discrete auction mechanism [6]
- Censorship resistance: No
  - TulipCross behaves as a dark pool of liquidity and thus it can be susceptible to censorship resistance in the same way as the lit exchange on which price discovery is carried out

8.3. **Limit Order Book (LOB).**

- Support for limit orders: Yes
  - Limit orders are naturally handled by a limit order book
- Welfare maximization: Medium
  - The presence of priority rules in the continuous matching might artificially reduce the achieved welfare of the participants
- Liquidity pooling across tokens: Low
  - LOBs only allow exchanging one type of token with another, artificially separating liquidity across different LOBs. Traders are thus forced to perform multiple trades when the desired token pair is not available, resulting in increased slippage and costs
- Liquidity pooling across time: High
  - LOBs let liquidity providers and takers trade against each other
- Trading efficiency: Medium
  - Although the continuous matching can be argued to be an efficient trading mechanism, it also enables predatory behaviors that reduce its efficiency from the point of view of non-HFT traders
- Transparency: No
  - The need to cancel and reprice orders, induced by continuous time, makes an LOB difficult if not impossible to run on a blockchain. Any solution that can not run on-chain does not guarantee transparency as required by decentralized finance.
- Impermanent loss: No
  - There is no impermanent loss, since liquidity is not locked inside the contract, contrary to the case for AMMs
- Custodial risk: Yes
  - For the same reasons described when reasoning about transparency, limit order books are run as centralized exchanges, which requires traders to delegate the ownership of tokens to centralized exchanges
- Rent extracted by high-frequency traders: High
  - Since time is treated as a continuous quantity, traders can gain advantages from faster connections and computation, allowing predatory behaviors from high-frequency traders [6]
- Censorship resistance: No
  - For the same reasons described when reasoning about transparency, limit order books are run as centralized exchanges, which requires

8.4. **Automatic Market Makers (AMM).**

- Support for limit orders: No

  - Uniswap V3 does not support limit orders directly, but only in terms of maximum slippage
- Welfare maximization: Small
  - In AMMs, price needs to be corrected by an arbitrageur, impacting the quality of the provided liquidity
- Liquidity pooling across tokens: Medium
  - Tokens are segregated in different liquidity pools, despite the role of base and quote tokens being interchangeable. Thus a swap between a generic token pair might require multiple swaps and fees
- Liquidity pooling across time: Low
  - Liquidity providers and takers interact with AMMs at different times and on different time scales
- Trading efficiency: Low
  - The matching is done between one liquidity taker at a time against liquidity providers, instead of letting traders interact directly
- Transparency: Yes
  - The implementation of an AMM can be done entirely on-chain and thus it guarantees transparency as required by decentralized finance
- Impermanent loss: No
  - AMMs are affected by impermanent loss, because liquidity providers and liquidity takers are in general different users operating at different time scales
- Custodial risk: No
  - Tokens are always under the control of their respective owners
- Rent extracted by high-frequency traders: High
  - In AMM price needs to be corrected by arbitrageurs in order to maintain alignment with large CEXs, where the price discovery happens, continuously creating arbitrage opportunities
- Censoriship resistance: Yes
  - The implementation of an AMM can be done entirely on-chain and thus it is robust to censorship

8.4.1. *Additional comments on AMMs.* Some of the benefits of AMMs are:

- conceptual simplicity
- low computational requirements
- ability to provide liquidity even for illiquid markets
- ability to function without a reference price

Some of the drawbacks of AMMs are:

- a rarely used building block used mainly in prediction markets rather than in mainstream finance. In fact only recently papers have started analyzing the financial return / risk profile of AMMs [16]
- liquidity providers are forced to trade at worse-than-market prices
- price needs to be corrected by an arbitrageur, impacting the quality of the provided liquidity
- artificially separates liquidity providers from traders, preventing traders from providing liquidity to each other in the way well-functioning markets do

Some critiques and counterpoints can be made to the benefits of AMMs listed above.

- **Conceptual simplicity**. Although this is a favorable point for users without experience in finance, the evolution of finance practices favors the use of limit orders, as explained in the introduction.

- **Low computational requirements**. This is not necessarily a strict requirement any more due to progress in off-chain computation and improved scalability in blockchain technology (such as layer 2 blockchains). We do not believe that an inefficient solution (from the point of view of exchanged value) should be preferred only because of implementation simplicity.
- **Ability to function even for illiquid markets**. Although this is a valid advantage for many markets (e.g., prediction markets), the vast majority of crypto coins are extremely liquid and do not require sacrificing trading quality.
- **Ability to function without a reference price**. This feature was certainly appealing to initial researchers (e.g., [7]) in search of a fully-decentralized solution to the problem of token exchange. In reality, the fact that price discovery unquestionably happens on current CEXs has turned AMMs into arbitrage generation machines (AGMs), where 80% of the trading volume is due to arbitrageurs keeping AMM prices in sync with the predominant price.

One problem AMMs address is that they provide a way for token holders to extract yield from their holdings. While this a useful function, new mechanisms are needed so that LPs may be compensated with revenues to offset the cost of their adverse selection, as quantified and characterized by [16].

References

[1] Hayden Adams, *Uniswap Whitepaper* (2018), available at `https://hackmd.io/s/HJ9jLsfTz`.

[2] Hayden Adams, Noah Zinsmeister, and Dan Robinson, *Uniswap v2 Core* (March 2020), available at `https://uniswap.org/whitepaper.pdf`.

[3] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson, *Uniswap v3 Core* (March 2021), available at `https://uniswap.org/whitepaper-v3.pdf`.

[4] Alejandro Bernales, Daniel Ladley, Evangelos Litos, and Marcela Valenzuela, *Alternative Execution Priority Rules in Dark Pools* (July 22, 2022), available at `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4169352`.

[5] Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould, *Trades, Quotes and Prices: Financial Markets Under the Microscope*, Cambridge University Press, 2018.

[6] Eric B. Budish, Peter Cramton, and John J. Shim, *The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response* (2015).

[7] Vitalik Buterin, *Let's run on-chain decentralized exchanges the way we run prediction markets* (2017), available at `https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way`.

[8] CME Group, *2023 FX Product Guide*, available at `https://www.cmegroup.com/trading/fx/files/fx-product-guide-2023-us.pdf`.

[9] ———, *Euro FX Futures - Contract Specs*, available at `https://www.cmegroup.com/markets/fx/g10/euro-fx.contractSpecs.html`.

[10] Robin Hanson, *Combinatorial Information Market Design*, Information Systems Frontiers **5** (2003), 107-119, available at `https://doi.org/10.1023/A:1022058209073`.

[11] Krishnamurthy Iyer, Ramesh Johari, and Ciamac C. Moallemi, *Welfare Analysis of Dark Pools* (2016), available at `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2040959`.

[12] Donald B. Johnson, *Finding all the Elementary Circuits of a Directed Graph*, SIAM Journal on Computing **4** (1975), no. 1, 77-84.

[13] Katya Malinova and Andreas Park, *Market Design with Blockchain Technology*, available at `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2785626`.

[14] Prabhaker Mateti and Deo Narsingh, *On Algorithms for Enumerating All Circuits of a Graph*, SIAM Journal on Computing **5** (1976), no. 1, 90-99, available at `https://doi.org/10.1137/0205007`.

[15] Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden, *Complexity-Approximation Trade-offs in Exchange Mechanisms: AMMs vs. LOBs* (2023), available at `https://arxiv.org/abs/2302.11652`.

[16] Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang, *Automated Market Making and Loss-Versus-Rebalancing* (2022), available at `https://doi.org/10.48550/arXiv.2208.06046`.

[17] Morgan Stanley, *Morgan Stanley Dark Pools*, available at `https://www.morganstanley.com/disclosures/morgan-stanley-dark-pools`.

[18] ———, *MS Pool ATS-N Filings*, available at https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&filenum=013-00117.
[19] Andreas Park, *Conceptual Flaws of Decentralized Automated Market Making* (2022), available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3805750.
[20] ———, *A 2022 Primer for Crypto-Trading* (2022), available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4148717.
[21] *Walrasian auction*, available at https://en.wikipedia.org/wiki/Walrasian_auction.
[22] Mu Xia, Jan Stallaert, and Andrew B. Whinston, *Solving the combinatorial double auction problem*, Journal of Operation Research **164** (2005), 239-251, available at https://www.sciencedirect.com/science/article/abs/pii/S0377221703008981.