DAOCROSS AND DAOSWAP

G.P. SAGGESE AND PAUL SMITH

Contents

1. Introduction

DaoCross and DaoSwap are smart contracts[1] that facilitate on-chain ERC-20[2] token exchange. DaoSwap provides a mechanism for efficient relative price discovery, and DaoCross enables parties to engage in zero market impact transactions with price improvement.

DaoSwap performs price discovery by matching supply and demand at regular time intervals. Supply and demand is expressed through a collection of buy and sell limit orders, and the clearing price is determined by a Walrasian-style auction ([11]).

DaoCross 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. Additionally, order intentions are not publicly disclosed prior to a cross, which enables block or other high-volume traders to execute quickly with minimal market impact.

DaoSwap and DaoCross differ in how the token exchange rate is determined, but both share several common advantages:

(1) Low Cost: they are a DeFi primitive that allows trading tokens peer-to-peer without incurring spread costs
(2) Capital Efficiency: participants provide liquidity only when they wish to trade, and only in the amount they wish to trade

Date: March 30, 2023.
With contributions from Tamara Jordania, Samarth KaPatel, and Daniel Yachmenev.

[1]https://ethereum.org/en/developers/docs/smart-contracts/
[2]https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

(3) 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
(4) No Intermediary Custody: exchanged tokens always remain under the control of their respective owners
(5) No Rent Extraction from High-Frequency Traders: speed alone does not confer an advantage to traders, as the discrete timing prevents predatory tactics such as front-running, limit order scalping, and spoofing, which are known issues seen with continuous limit order books
(6) Ease-of-use: the interaction between buyers and sellers occurs through smart contracts, with a website front-end available

2. Matching liquidity

Suppose there are two parties, Alice an ETH holder and Bob a wBTC holder (wrapped bitcoin, an ERC-20 compliant coin that tracks bitcoin), who wish to = swap tokens at a competitive rate. Suppose also that there are many additional ETH and wBTC holders who may be interested in participating in a swap. 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 ETH and 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 will consider and address these in the sequel, but to begin, we will discuss a more constrained setting.

2.1. **Breaking the symmetry.** To simplify, we follow the practice of FX (foreign exchange) futures markets, which break the symmetry 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, [6] (e.g., footnotes on page 25) for usage of this terminology. See [7] 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 form, 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 the same.

2.2. **Basket case.** 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 contrive 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 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

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., [12]), which is a significant limiting factor in terms of feasibility and auction expense.

3. Protocol Introduction

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).

3.1.1. *Example of base/quote tokens.* By way of example, 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")

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.

3.2. **Order attributes.** A DaoCross or DaoSwap order must have the following attributes:

(1) Base token
(2) Quote token
(3) Action (buy/sell)
(4) Quantity
(5) Limit price
(6) Timestamp
(7) Deposit address

We have discussed attributes 1-5 in some detail.

Timestamp (6th attribute) is used to determine eligibility in a swap or cross and can play a role in order fill priority.

Deposit address (7th attribute) enables one to use multiple wallets to facilitate account management. 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. This functionality is also standard in traditional finance, e.g., purchasing T-Bills on TreasuryDirect.

3.2.1. *Order notation.* We introduce the following tuple notation for a general limit order:

```
(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` a `quantity` of the token `base_token` for a limit price of `limit_price` with respect to token `quote_token` and deposit the resulting tokens at `deposit_address`."

For brevity, in the sequel we may:

- omit the timestamp and the deposit address when not relevant
- omit the (infinite) limit price for market orders

3.2.2. *Example of 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 buying up to 3.2 units of ETH in exchange for wBTC up to a price of 4.0 wBTC per ETH with proceeds deposited at `0xdeadc0de`".

3.2.3. *Example of market order notation.* Consider a swap for the tokens ETH, wBTC and assume that:

- the exchange rate between ETH and wBTC is 0.2 (i.e., 5 ETH can be exchanged for 1 wBTC and vice versa)
- there is no constraint on the quantity of ETH and wBTC available for each trade participants

Then

- An order `(buy, 1.0, ETH, nan, wBTC)` means buying 1 ETH in exchange for 0.2 wBTC
- An order `(sell, 1.0, ETH, wBTC)` means selling 1 ETH, receiving the corresponding amount of 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 exchanging 1 wBTC with 5 ETH

3.2.4. *Example of limit orders.* Consider the same swap for ETH and wBTC above, assume the same exchange rate of ETH and wBTC, i.e., $\frac{p_{ETH}}{p_{wBTC}} = 0.2$, assume that each agent has a very large amount of ETH and wBTC.

A limit order `(buy, 1.0, ETH, 2.0, wBTC)` means "buy 1 ETH paying in wBTC assuming that the price of 1 ETH is at most 2 wBTC". In this case, since the price of one ETH is equal to 0.2 wBTC, the exchange can be executed with another party.

On the other hand, a limit order `(buy, 1.0, ETH, 0.1, wBTC)` requires that the price of ETH be lower than the current price and this does not allow the token swap to be carried out.

A limit order `(sell, 1.0, ETH, 2.0, wBTC)` means "sell 1 unit of ETH, receiving wBTC as long as the price of one unit of ETH is more than 2 wBTC". Because $p_{ETH} = 0.2 \cdot p_{wBTC} < 2.0 \cdot p_{wBTC}$, the order can be executed.

3.2.5. *Limit order as a set of linear inequalities.* Any limit order as defined above can be translated into inequalities involving quantity of exchanged tokens and allowed token prices. These equations can then be collected in a system of equations governing the equilibrium achieved by the orders.

We indicate with an asterisk the quantities resulting from an executed swap order:

- $q^*_{base}(o_i)$ the actual quantity of the base token exchanged in the swap requested by order o_i
- $p^*_{base}(o_i)$ the actual price of the base token exchanged as per o_i
- Same definitions hold for the quote token q^*_{quote} and p^*_{quote}

An order like $o_1 =$ `(buy, 2, A, 3, B)` means "buy 2 units of token A paying with token B with a limit price for A of at most 3 B" and it corresponds to a set of inequalities on the actual quantity of A obtained and the price paid:

$$\begin{cases} p^*_{base}(o_1) = p_A \leq 3p_B \\ (0 \leq q^*_A(o_1) \leq 2) \wedge (p_A \leq 3p_B) \vee (q^*_A(o_1) = 0) \end{cases} \tag{1}$$

The constraint on the quantity exchanged is conditioned to the corresponding price satisfying the desired limit price constraint, otherwise the swap can not be carried out and the exchanged quantity is 0.

While the constraint on quantities is specific of each order, the constraint on price of base / quote token is a global one on the price of the corresponding token since the price of a token needs to be the same across all the swaps.

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

$$\begin{cases} p_A \geq 2p_B \\ (0 \leq q^*_A(o_2) \leq 3) \wedge (p_A \geq 2p_B) \vee (q^*_A(o_2) = 0) \end{cases} \tag{2}$$

3.2.6. *Market order as a set of linear inequalities.* A market order (i.e., without a limit order) has no constraint on price and thus it is reduced to:

$$\begin{cases} p_A \leq 3p_B \\ 0 \leq q^*_A(o_1) \leq 2 \end{cases} \tag{3}$$

3.2.7. *Order equivalence.* If the exchange rate between two tokens A and B is known and equal to $p_A/p_B = c$, the four types of orders corresponding to "buying" vs "selling" and to "A for B" vs "B for A" can be reduced to only two types since a buy (sell) order for token A is equivalent to a sell (buy) order for token B after properly converting the quantities and the limit price given the value c.

This is easily shown in terms of the corresponding inequations describing the orders:

3.2.8. *Example of order equivalence.*

3.3. **DaoCross reference price.** DaoCross 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]).

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 BTC/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.

3.4. **Lit 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

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

expressed in BTC 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, DaoCross 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:

$$\frac{\text{bid}_{\text{quote token}} + \text{ask}_{\text{quote token}}}{\text{bid}_{\text{base token}} + \text{ask}_{\text{base token}}} \tag{4}$$

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 so as to avoid extreme price variations and to mitigate the risk and effects of any market manipulation attemps.

3.4.1. *Formulation of the general DaoSwap problem.* The general problem of determining the token equilibrium price 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 is preserved across the swaps, i.e., the quantity bought for each token is equal to the sold quantity.

Consider a set of limit orders in the form

$$o_i = (a, q_{base}, lp_{base}, q_{quote} = (a(o_i), q_{base}(o_i), lp_{base}(o_i), q_{quote}(o_i))$$

We encode the direction of the inequality on the price in a function:

$$to\_ineq(a) = \begin{cases} +1 \text{ for } a = \text{buy} \\ -1 \text{ for } a = \text{sell} \end{cases}$$

Assume by convention that a buy order removes base tokens from the system and provides quote tokens to the system, whereas a sell order does the opposite.

The sign of the contribution of base tokens by an order is encoded in

$$to\_diff(a) = diff_{base} = -diff_{quote} = \begin{cases} +1 \text{ for } a = \text{buy} \\ -1 \text{ for } a = \text{sell} \end{cases}$$

E.g., if o_i is a buy order and it is executed in its entirety, then o_i takes away from the system $q^*_{base}(o_i)$ tokens of type $to\_token(q_{base}(o_i))$ and provides $q^*_{quote}(o_i) = q^*_{base}(o_i) \cdot \frac{p^*_{quote}}{p^*_{base}}$ tokens of type $to\_token(q_{quote}(o_i))$.

Note that we could have express $to\_ineq(a)$ in terms of $to\_diff(a)$ but we keep these two auxiliary variable separated for sake of simplicity.

Each order imposes the constraint:

$$\begin{cases} cond_i : to\_ineq(a(o_i))(p_{base}(o_i) \leq lp_{base}(o_i)p_{quote}(o_i)) \\ (0 \leq q^*_{base}(o_i) \leq q_{base}) \wedge cond_i \vee (q^*_{base}(o_i) = 0) \\ exch\_q_{base}(o_i) = to\_diff(a(o_i)) \cdot q_{base}(o_i) exch\_q_{quote}(o_i) = -to\_diff(a(o_i)) \cdot q_{base}(o_i) \cdot \frac{p_{quote}}{p_{base}} \end{cases} \tag{5}$$

Rewriting all the constraints in terms of orders and the unique set of base and quote tokens: $to\_token(q_{base}(o_i)) = t_1, ..., t_n$

$$\begin{cases} \forall \text{ order } icond_i : to\_ineq(a(o_i))(p^*_{base}(o_i) \leq lp_{base}(o_i)p^*_{quote}(o_i)) \\ \forall \text{ order } i(0 \leq q^*_{base}(o_i) \leq q_{base}) \wedge (cond_i) \vee (q^*_{base}(o_i) = 0) \\ \forall \text{ order} iexch\_q_{base}(o_i) = to\_diff(a(o_i)) \cdot q_{base}(o_i) \\ \forall \text{ order} iexch\_q_{quote}(o_i) = -to\_diff(a(o_i)) \cdot q_{base}(o_i) \cdot \frac{p_{quote}}{p_{base}} \\ \sum_{t_j} exch\_q_{base}(t_j) = 0 \\ \sum_{t_k} exch\_q_{quote}(t_k) = 0 \end{cases} \tag{6}$$

TODO(gp): Check the math better by implementing it. Also try to simplify the formulation which is still cumbersome.

3.4.2. *Solution of the general DaoSwap problem.* The general solution of the DaoSwap problem is a set of non-linear equation with combinatorial constraints, which is of course NP-complete.

It can still be solved in an effective way with solvers like ...

TODO(gp): explain better, cite

3.4.3. *Formulating DaoCross problem.* In the DaoCross set-up, the price of the tokens involved into the swap is determined by an external oracle. This allows to simplify the general problem by applying the equivalence principle between orders and removing the non-linearity from the set of inequalities above.

In fact the actual prices p^* compared to the limit price verifies or not the boolean condition $cond_i$ which then allows to specify the quantity constraints:

$$\begin{cases} cond_i := to\_ineq(a(o_i))(p_{base}(o_i) \leq lp_{base}(o_i)p_{quote}(o_i)) \\ 0 \leq q_{base}(o_i) \leq q_{base} \text{ if } cond_i \\ q_{base}(o_i) = 0 \text{ if } \neg cond_i \end{cases} \tag{7}$$

The problem then becomes a set of linear inequalities that can be solved with several efficient methods (e.g., simplex-method).

TODO(gp): check

3.5. **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 [10] for the prioritization rules used by one of Morgan Stanley's equity liquidity pools.

3.5.1. *Priority rules.* DaoCross 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.

3.5.2. *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. One an order is fully filled, it is removed from the priority queue. The procedure continues until one of the two priority queues is empty.

3.5.3. *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, order removal costs $O(n \log n)$, and order remove occurs $O(n)$ times. So, the computational time complexity of the task is $O(n \log n)$. Memory requirements are $O(n)$.

3.5.4. *DaoSwap variant.* The mechanics for DaoSwap are similar to those above, with the primary difference being that an auction is used to determine a single clearing price. Variations in prioritization rules also possible.

3.6. **Fees.**

3.7. **Comparison with Uniswap.** Uniswap doesn't support limit orders - v3 introduced some steps towards incorporating - DaoSwap/Cross supports limit orders

Because liquidity providers and liquidity takers are in general different users operating at different time scales, Uniswap is affected by impermanent loss - DaoSwap/Cross is not affected by that

Uniswap requires multiple swaps and fees for arbitrary tokens - DaoSwap/Cross pools all the liquidity in a single optimization problem

4. Example

References

[1] Hayden Adams, *Uniswap Whitepaper* (2018).

[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* (2018).

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

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

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

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

[10] ———, *MS Pool ATS-N Filings*, available at `https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&filenum=013-00117&owner=exclude&count=40`.

[11] *Walrasian auction*, available at `https://en.wikipedia.org/wiki/Walrasian_auction`.

[12] 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?via%3Dihub`.