In our view batch auctions are in interesting direction for DEX research. In contrast to “normal” orderbook (continues double auction) dexs where trades are executed continuously, batch auctions collect orders into a batch (discrete time) and then calculate a “single clearing price” and which all orders are executed.

If you do not want to rely on a “operator”/“trade coordinator” or any other central entity that creates an order of transactions/orders the smalest unit of time is “one block”. This is - you can say this transaction came first if it is part of an earlier block, but if 2 transactions are part of the same block there is no way to say which came first.

Exchanges like OasisDEX or Uniswap however nevertheless treat orders in the same block differently depending on the order in which the miner includes them into a block. Ultimately this is completely arbitrary and can be exploited by miners and adds to what @phil calls “miner extractable

value” (MEV)— here.

Our simple claim is:

The same 2 orders in the same block should get the same price.

In a batch auction a batch could be potentially as small as a block. This would reduce the MEV but miners would still have significant influence by potentially censoring specific orders. Thus it might make sense to have a batch running over multiple blocks. In any case - we need to define a strategy of how to “clear a batch” == “find the single clearing price and settle the orders accordingly”.

A paper that argues for batch auctions for different reasons describes a clearing strategy as follows:

Simplified speaking - either there is no overlap in the order book - than no matches, or if there is overlap, than where supply and demand cross.

More exact version here:

Case 2: Supply and demand cross. Typically, if supply and demand cross they do so horizontally. Let p*

denote the unique price and let q denote the maximum quantity. In this case, all orders to buy with a price greater than p*and all orders to sell with a price less than p* transact their full quantity at p. For orders with a price of exactly p* it will be possible to fill one side of the market in full whereas the other side will have to be rationed. We suggest the following rationing rule: pro-rata with time priority across batch intervals but not within batch intervals. That is, orders from earlier batch intervals are filled first, and in the single batch interval in which it is necessary to ration, this rationing is pro-rata treating all orders submitted within that interval equally. This policy encourages the provision of long-standing limit orders, as in the CLOB, but without inducing a race to be first within a batch interval. There is also a knife-edge case in which supply and demand intersect vertically instead of horizontally. In this case, quantity is uniquely pinned down, and the price is set to the midpoint of the interval. There is no need for rationing in this case.

Now this assumes a case of a single trading pair. There are plenty of reasons to allow many assets/tokens be traded all in one system. A primary reason is that you can allow so called “ring trades”. If trader 1 wants to trade A->B, 2 B-C and 3 C-A those 3 trades could match together.

To illustrate this: those are the trading pairs on Kraken and since many tokens are traded in several currencies/tokens there is plenty of opportunity for “ring trades”.

[

08

1148×1024 258 KB

](https://ethresear.ch/uploads/default/original/2X/f/ff50c937f6718c3f643755c3bf902029356b27c9.jpeg)

However - in this multi asset world the question how to settle the trades becomes much less obvious.

Since we want an arbitrage free result prices of A<->B and B<->C should also define the prices of A<->C.

A very simple implementation to ensure this is to store just one price number for very token and the prices of every pair are calculated by the ratio of the price number of 2 tokens. Obviously that means that if you optimizing for the A<->B pair and you are changing the price of A you are influencing all trading pairs that contain A.

A valid solution in addition to the price consistency need to also full-fill “value preservation” - that is, for each token the total number sold and bought need to equal out.

Here is a visualization of optimizing prices for 3 tokens.

[

25

1102×501 122 KB

](https://ethresear.ch/uploads/default/original/2X/4/459aecc4b2df7b6dad75b99e5604ac961f01afed.png)

When it comes to this optimization problem we currently have the assumption that this is a NP-hard problem and thus the optimal solution can not practically be calculated deterministically (and certainly not on the EVM). We want to deal with this by creating a open competition. After a batch closes anyone can submit a solution within a fix time period and the best on is executed. In this case the 2 operations that need to be done on-chain are simply: make sure the solution is valid; check the value of the objective function.

The problem now is: what is a good objective function.

1. Overall trading volume (all trades would need to be converted based on current prices to a reference token)
2. Overall trader utility (simply speaking - (how much of an asset did a trader get - how much did they wanted to get at least)

We can shorty summarize the constraints of a solution:

1. Pick a price for each token
2. Orders that are on or below the clearing price CAN be executed
3. Define a volume (from 0-100%) for each order that CAN be excuted
4. Make sure that for each token “total amount sold == total amount bought”

Let’s create a simple example and see how those two objective functions perform.

"Volume creates unnecessary long ring trades"

Example volume optimization

:

Consider we have 3 stable coins that should all be worth $1. (S1, S2, S3) We have 3 market makers that each have 1000 of one of the stable coins and they are willing to trade them for every other stable coin if they make a spread of at least 1% on the trade. If those 9 trades where the only open orders in the system no trade would be possible.

Now a user comes that simply wants to convert 10 S1 to S2. They post a trade of e.g. 10 S1 for at least 9.98 S2.

A system that will optimize for volume will now set prices of S1 to 1.00 of S2 to 0.98 and of S3 to 0.99. Now it can execute a full ring. The user Trades S1 to S2, market maker 2 trades S2 to S3 and MM3 trades 3 to 1. Bother market makers make 1% on the trade and the user “pays 2 %”. The overall trading volume is roughly $30.

For further discussion, please note that MM2 is trade S2 for S3 (at there limit price) but the order S2 for S1 which would create more utility for MM2 remains full unfilled although the limit price is strictly below the clearing price.

So the intuitive solution to simply settle the trade of the user with the market maker that is willing to do the opposite trade does NOT maximize volume. If the user would pick a lower limit price and the number of tokens would be increase (possibility for bigger rings) the problem would get worse. Basically the user is exploited to the max to allow more trades.

Example utility optimization

:

The described problem can be fixed if we optimized for “overall trader utility” instead. Utility in the example above would be 0 since all trades are exactly executed at the limit price == you could argue the price at which the person is indifferent between trading and not trading. However - any solution that lets MM2 trade directly against the user will create utility for MM2 if executed at a price of 0.998 for the user if executed at 0.99 and for both if executed anywhere in between. If writing and optimizer it turns out that the optimal solution will split the utility 50% 50% between the two trades. This might lead to an “acceptable price” in this example of ~0.9985 but it gets worse if the user would submit a “market order” == “an order with a very low limit price”.

We would argue if the market provides plenty of liquidity at $0.99 then the user should get that price. Of course the system does not differentiate between market maker and user, so simply the fact that more liquidity is available at a price makes this price the market maker price.

Those prices can be enforced if we tighten the rules for a valid solutions:

1. Pick a price for each token

2. …

a) Orders that are below the clearing price HAVE TO BE FULLY executed

b) Orders that are on the clearing price CAN be executed

1. Define a volume (from 0-100%) for each order that CAN be executed
2. Make sure that for each token “total amount sold == total amount bought”

Now we still could not come up with a proof (nor a counter proof) that a valid solution under those rules is always possible. But we are pretty certain that there is no way that a valid solution can be practically found in a given time period.

For this reason we tried a 3rd optimization criteria that does not fully forbid unfilled orders below the “clearing price” but it penalized them. We called it “disregarded utility”. Basically for each order it calculates the utility as before but IN ADDITION it subtracts the amount that was “disregarded”. “Disregarded utility” is defined here as the utility that would have been generated if the order was executed fully - how much utility was actually generated. If the clearing price is == the limit price of an order the utility is always 0 no matter how much was executed. However - if the limit price of an order is below the clearing price it will generate disregarded utility UNLESS it is executed fully.

So far this optimization metric has generated results that are in line with what we intuitively would have expected the system to do.

Solution verification on-chain

The only big downside of this metric over utility and volume is that the solution verification/ specifically the measurement of the objective function are now in terms of gas costs depended on the total number of open orders instead of only those orders that are executed in the batch.

This is a reason why we are looking for other objective functions where ideally only the touched orders play a role in the objective function.

Additional resources:

GitHub

gnosis/dex-contracts

Smart contracts for dƒusion batch auction exchange - gnosis/dex-contracts

Smart contract that allows so far submissions optimized for volume

Paper on how to convert the optimization problem for volume into a mix integer optimization problem.

batchauctions_or2018.pdf (187.9 KB)