One of the main goals of CoW Protocol is to allow users to trade at a fair price. Therefore, all solvers participating in the solver competition need to provide fair prices to users. This was established in CIP-11 as a social consensus rule. The documentation gives a more formal interpretation of what a fair price is and refers to it as Ethereum's Best Bid and Offer (EBBO).

We have observed in the past that not all settlements conform to that rule due to, for example, how surplus is shared among its various traders. The goal of this post is to start a discussion around how to get to a point where all executed trades satisfy EBBO.

What is EBBO?

The term Ethereum's Best Bid and Offer (EBBO) was coined in analogy to National Best Bid and Offer (NBBO). The latter is a rule in traditional finance protecting retail traders. There, brokers have to provide best deals to traders by looking at prices on multiple exchanges, splitting the traded volume across exchanges if necessary. The same idea applied to the Ethereum ecosystem is EBBO.

One main question is about which exchanges need to be considered for EBBO. The documentation specifies those exchanges as base liquidity by defining base protocols and base tokens. Then any liquidity source from those base protocols connecting a trader's sell token to the buy token, sell token to a base token, and base token to a buy token needs to be considered by solvers.

[

EBBO routing

1211×720 82.9 KB

](https://europe1.discourse-cdn.com/business20/uploads/cow/original/2X/8/83efa5026abfc7c5266a5027b42020ecdc58f659.png)

Choosing only the most popular protocols as base protocols restricts the types of exchanges to consider. The inclusion of base tokens does account for the fact that a lot of liquidity is provided via WETH or stable coins, and not always via an immediate connection from the sell to the buy token. Thus exchanges have to be combined, with a maximal path length of two.

The description of EBBO above is, however, not a precise definition. One reason for this is changes in pools during the cause of an auction. Another reason is batching of solutions which can lead to negative externalities. More on that later, but first an example.

Example

Suppose a user wants to sell 100 WETH to buy COW tokens on Ethereum mainnet. Further suppose there are the following AMMs:

- a constant product pool on Uniswap with reserves of 100 WETH and 1M COW
- a constant product pool on Balances with reserves of 100 WETH and 1k GNO
- a constant product pool on Sushiswap with reserves of 1k GNO and 1M COW
- there are no other pools connecting COW to a base token

[

EBBO routing example

1201×493 55.1 KB

](https://europe1.discourse-cdn.com/business20/uploads/cow/original/2X/6/6f4fb2105673f428ee16c0139099932b69389a98.png)

The three pools mentioned are baseline liquidity since they are pools by base protocols and are connecting sell and buy token, sell and base token, and base and buy token. No other pools are baseline liquidity. For simplicity, the execution cost of the settlement will be 0.01ETH plus 0.01 ETH per AMM used.

The optimal routing splits funds across all three pools. About two thirds should be routed through the first pool and one third through the other two pools. This results in around 600k COW tokens.

A sophisticated user will be able to split funds accordingly or just use a router like 1inch which will easily find such a route. Therefore all solvers on CoW Protocol should be able to provide at least 600k COW to the user, independently of how those

funds are provided.

Routing all funds through the first pool will result in around 500k COW tokens, routing everything through the latter two pools in around 333k COW tokens. Both of those routes do not provide EBBO as they provide less COW tokens than the simple split across three pools.

Current state

Not all solvers explicitly enforce EBBO in their solutions. In almost all cases, this is not

an issue since the solver competition ensures that better executions of trades are incentivized.

There are mainly two cases where EBBO violations occur.

1. If there is little competition in an auction.
2. If there is batching.

In the first case, a solver might just win with a suboptimal solution due to a lack of competitors with better solutions. Since CoW Protocol is a real time system, this case will happen from time to time, e.g. due to technical issues. It is important for the protocol to keep working in such cases and to keep providing users with executions which are at least as good as what they would get elsewhere. This becomes even more relevant when more chains are being supported by CoW Protocol which will start out with fewer solvers.

In the second case, solvers can win the competition by providing a lot of surplus to one user even if they give a subpar execution to another user. Apart from outright sub-optimal routing, this can also happen if batch fees are not distributed fairly across orders. This is not uncommon in practice, albeit only for small amounts. Another prototypical example is shifting surplus from some users to others. Even though there is an incentive to provide better routing in batches, distribution issues are not currently prevented by the incentive mechanism.

Making EBBO a strict rule

EBBO is notoriously difficult to define in a formal manner. We still want to move in the direction of making it a hard rule of the competition since it guarantees that users of CoW Protocol are protected from getting suboptimal executions. It will also make it easier for solvers to follow EBBO rules.

EBBO as a special solver

The EBBO for an order is the best price it can get using a specified set of liquidity sources. This means that it is defined via the solution of an optimal routing problem. Finding optimal routing is precisely one of the tasks of a solver in the solver competition. Defining and checking EBBO thus means having access to a solver. In the following we call such a solver EBBO solver.

Efficient algorithms have been proposed for such problems, see here. Even though computing optimal routes is not a trivial task it can be done for individual orders in significantly less than one second, given the state of liquidity sources to be used. Note that the EBBO solver for the routing problem is less complicated than a solver capable of batching solutions.

To ensure transparency of processes around EBBO violations, the EBBO solver would need to be open source and easily runnable by external parties. It should be maintained by the CoW Protocol community and development team.

EBBO violations

Using the EBBO solver we can define what constitutes an EBBO violation. A trade violates EBBO if the EBBO solver can provide more surplus to the order. In such cases, traders need to be reimbursed by the solver who executed the sub-optimal settlement. The reimbursement should cover the difference in surplus when comparing EBBO to the actual settlement. Thus all users are protected to get at least EBBO when using CoW Protocol.

Challenges and corner cases

There are various challenges in defining and enforcing EBBO.

The most common cause for false positives in detecting EBBO violations in the paste were large shifts in pool states between when the liquidity for an auction was collected and when solvers submitted their solution. A route which is optimal when submitted to the Protocol need not be optimal when executed. This also makes it problematic to use quotes or the quote competition as a proxy for EBBO. One way to address this is to run the EBBO solver for liquidity fetched at different blocks between cutting the auction and the submission on chain. We also need to make sure via simulations that the solutions by the EBBO solver are executable on chain.

Another major issue comes from setting user fees by estimating execution costs and revert risk. An EBBO solver needs to be able to set reasonable fees, which goes beyond a simple routing problem. A conservative approach to fees and solver parameters chosen by the community should give a reasonable estimate for EBBO.

Structurally, the biggest difference between EBBO and NBBO comes from the possibility of batching. It is not in all cases possible to satisfy EBBO for individual orders in an optimally batched solution. As an example, think of multiple trades on the same token pair in the same direction resulting in additional price impact if executed simultaneously. In most cases those effects should be small compared to efficiency gains from batching. If those effects turn out to be significant, EBBO needs to be adapted to take batching into account. We could also introduce a mechanism for challenging an EBBO violation to mitigate such issues.

The EBBO solver will not be able to catch bad executions for exotic tokens requiring additional hops, e.g., due to wrapping and unwrapping. The open source nature of the EBBO solver would, however, allow for adding additional routing options if deemed useful by the community. This could evolve into an EBBO liquidity book where additional paths/sources could be added decentrally.

Keeping the EBBO solver up to date and supporting new features of the protocol will require spending time on maintenance. Adding new features must not increase run times by much since the solver needs to be able to participate in the real time competition.

A first proposal

- An open source EBBO solver is implemented. It needs to be fast enough to compute an execution conforming to EBBO for individual orders in less than a second.
- The EBBO solver checks every trade in each settlement which is executed on chain. If it can provide more surplus to an order on all blocks between order creation and the settlement on chain, the settlement violates EBBO.
- In case of an EBBO violation, the winning solver needs to reimburse affected users for the difference in surplus of what EBBO provided and what their solution provided.

Next steps

At the moment we are lacking visibility into EBBO violations. There have been EBBO violations in the past and users were reimbursed. But the current monitoring does not reliably identify EBBO violations. Building a prototype for an EBBO solver would allow us to get an overview of EBBO violations. The solver could be built by the core team or via a grant of the CoW DAO by a community member or one of the current solvers.

While we get an overview over EBBO violations, we will work with solvers on improving the quality of their executions to minimize impact on users.

After testing the EBBO solver, a formal EBBO test as proposed above would be enforced as a rule to the solver competition.

Alternative approaches to ensuring fair execution of trades should also be discussed in case testing with an EBBO solver is not successful.