La vache qui Vickrey

This is a proposal for a new auction mechanism for the solver competition. On a high level, it looks like a Vickrey Auction, adapted to solver competition. The primary motivation is that it makes it unprofitable for solvers to engage in the type of strategic behavior discussed in this thread. A secondary motivation is that it

allows rewards to be determined by the market based on how risky and difficult it is to find good solutions for a batch. This means the tokens paid by the DAO to incentivize solver development will more efficiently spent.

What is a Vickrey auction?

In a regular Vickrey auction, also called second-price auction, a seller auctions off an item to multiple buyers. The buyers each submit a sealed (meaning hidden from the other bidders) bid for the item. The bids are ranked from high to low, and the highest bidder wins the item. This bidder then pays not the price it bid, but the bid of the second highest

bidder. Eg if there are bidders bidding $10, $12 and $15, the last bidder wins and pays $12.

While this may seem weird on first sight, it has the positive side effect that the optimal strategy for buyers is simple: just bid

how much the item is worth to you. It is incentive compatible

in the sense that bidders are incentivized to bid what they truly think the item is worth. If the seller were to charge the bid price to the winner, this would no longer be true. The buyer has an incentive to try to 'guess' what the next highest bidder would bid, and then bid exactly 1ct more. To avoid this kind of complexity on the buyer's side, the second-price auction is a great solution. And, under some mild assumptions,

charging the second price auction will yield the same revenue as the first price auction on average (and the same revenue as a whole bunch of other auction mechanisms, a consequence of so-called Revenue Equivalence Theorem).

A Vickrey mechanism for the solver competition

The second price auction doesn't directly apply to the solver competition. Firstly because it's more like a reverse auction: cowswap acts like a buyer and solvers are competing to sell the best solution at the lowest price. Secondly, because there isn't one well defined item. Different solvers can submit solutions with different utilities to the protocol. Lastly, we want to make sure that we don't commit to paying an unbounded amount of reward to the winning solver. The solution proposed here

addressed all these issues in a way that feels aligned with the general objectives of the protocol in my opinion.

Mechanism description

Caveat: for this to work we need to change the slippage rules to be symmetric. That is, solvers get negative slippage subtracted from their reimbursement and positive slippage added to their reimbursement. If we keep using the one-sided version we use now there is still some room for strategic behavior, in terms of making sure that the slippage is as close as possible to 0 at the end of the week

Like the status quo, every solver submits a solution for which we compute the total utility as surplus minus estimated gas cost plus fees received. However, solvers now additionally submit a 'price', representing the minimum cow reward they need to cover the cost of failed transactions, net slippage and any other cost they might occur. Note that this number may be positive if there is a strong expectation of positive slippage.

The solvers are now ranked by utility - price, and the

highest value is awarded the batch. If this solver succeeds in executing the batch, they are reimbursed for their gas cost adjusted for net slippage. Additionally, they receive a cow reward equal to their price plus the difference in the objective function between the winner and the second highest solution

. Note we don't just look at the price reported by the solvers, as the utility of the solutions might be different and we need to adjust for that.

Strategy proofness

I haven't worked it out formally, but I believe this mechanism is strategy proof. That is, it's optimal for a solver to report their solution exactly as they believe it will play out in the typical case and then provide a bid equal to their expectation of revert cost + average slippage.

The argument is something like, if you try to penny to improve your objective function you might as well adjust your price by

the same amount, it doesn't matter for your rank and the cost is cancelled out by the price, so reporting true values for the token flows is optimal. But then the only thing to play around with is price, and given a fixed solution, changing the price doesn't affect the total reward if you win, only whether you win or not. In other words, we have the same dynamics as the second price auction.

Reserve price

To ensure we don't pay too much rewards when a solver happens to find an insanely good solution, we might cap the rewards for the batch (the reserve price

). This means that a solver never receives more reward than the reserve price, even if the difference with the next best solution is very large.

On submitting the best solution

The mechanism without reserve price is also strategy proof in the sense that solvers are incentivized to report the best solution they have found in all cases. By introducing a reserve price there may be edge cases where a solver could report a slightly worse solution that has lower risk if they expect the competition to be particularly week.

Example: let's say the reward is capped at 100, and the solver has found a solution with 200 utility for which they expect a cost of 10 on average, and a

solution with 100 utility for which they expect a cost of 0. Suppose they expect the competition to find only solutions with utility 0 or less and positive price, then they are better off reporting the worse solution, so they can get the full 100 reward and 0 cost, instead of the reward capped at 100 at 10 cost.

This problem equally occurs with the current mechanism though, so it would not be worse than the status quo. I suspect this will be too complex and too rare to exploit for the foreseeable future anyway.

Some examples

I just describe some two solver examples here for simplicity. It is enough to illustrate the full extent of the mechanism

Example 1

Suppose the open orders contain only a single, small stable to stable trade, and both solvers find very similar solutions.

Solver 1 finds a solution with utility 1, fee 10 and gas cost 10, and quotes a price of 1 to cover losses. Solver two finds a solution with utility 2, fee 10 and gas cost 10, but quotes a price of 1.5. This means the objectives are:

- solver 1: 1 - 10 + 10 - 1 = 0
- solver 2: 2 - 10 + 10 - 1.5 = 0.5

Solver 2 wins the batch. If the settlement is successful, they get a reward equal to 1.5 (their price) + 0.5 (the difference with the second best solver) for a total of 2.

Example 2

Similar situation, but now we are in a high volatility period meaning gas prices and revert probability are high. Solver 1 finds a solution with surplus 1, fee 100 and gas cost 100, and quotes a price of 10 to cover losses (higher now because of high gas prices and revert risk). Solver 2 finds a solution with surplus 2, fee 100 and gas cost 100, but quotes a price of 10.5. This means the objectives are:

- solver 1: 1 - 100 + 100 - 10 = 0
- solver 2: 2 - 100 + 100 - 10.5 = 0.5

Solver 2 again wins the batch. If the settlement is successful, their reward is 10.5+0.5 = 11. Note that their margin is probably still quite small, reflecting the fact that this was a very competitive batch and they didn't bring that much incremental value. The high reward is purely to compensate for the execution risk, not the quality of their solution.

Example 3

Now look at a situation where one solver submits a solution that is superior in every way. Just to make things a bit more real, let's base it on this particular settlement that actually took place:

etherscan cow explorer

The solver that won this batch settled both orders through 0x for a surplus of roughly $90 and $100 respectively and a total gas cost of around 100. In principle anyone could have found this solution and competition was probably fierce. Let say this was the solution of solver 1. Now imagine that another solver had detected that the both paths through 0x ran through the 0.5% fee UniV3 USDC/ETH pool in opposite directions. In principle the entire order selling 9.5 ETH for USDC could have been settled

directly against the USDC/ETH leg of the other order. This would have saved a cumulative volume of around 21K USDC running through uniswap, saving about 105$ in fees. In addition, it would save some gas cost as crossing ticks costs gas in UniV3, and it likely wouldnt' have been necessary to split the USDC/ETH leg of the other order across multiple sources anymore saving further gas. Let's say 30$ for the sake of argument. Let's assume the fees were also about 100$ and both solvers required a

price of 10. The objectives would look like this:

Solver

- solver 1: 190 - 100 + 100 - 10 = 180
- solver 2: 290 - 70 + 100 - 10 = 310

Solver 2 wins the batch and receives a reward of 10+130=140. Note that finding this solution is an engineering problem that could be solved, but it's a lot of work and it only helps to win rare batches. More work not only means engineering hours that can't be spent improving algorithms for easy batches, but also compute used in the inner solver loop that can't be spent looking for marginal improvements that occur more frequently, rate limits that could be used searching for different types of solutions etc.

If solvers are rewarded for the incremental value they deliver in such cases we might see more resources spent on chasing these high value/low frequency improvements.