Operator Pool Trees: filling order flow ASAP—at least in theory

If you (at some point in your life)

> criticized the broad use of mathematical models for human behaviour, arguing that some human choices are irreducible to mathematics
> [Wikipedia: Mathematical Economics]

you will have a field day criticizing this blog post. If not, I hope you agree with my strategy in writing this article:[1] I will start by presenting the whole background, in particular utility functions, discount factors, and social welfare optimization (in the hope that the critics get too bored), before the presentation of the core idea: a model for networks of interconnected frequent batch auctions of operator pools that allows to rule out certain constellations, because they are utter nonsense (for the purpose of welfare maximization)—even under the most favourable conditions. Finally, I conclude with a tentative research agenda for how the insights gained could be a basis for *distributed* frequent batch auctions as part of Anoma as an intent-machine for Ethereum. So, buckle up for a *»tour Xe force«* through some *elements of mathematical economics* or skip to the end for how this relates to Anoma as an intent machine for Ethereum.[2]

Utility functions, discounted utility, & maximal welfare

Utility functions and the Von Neumann–Morgenstern utility theorem

Wikipedia's short explanation of utility function is that[3]

> a decision-maker faced with risky [...] outcomes of different choices will behave as if they are maximizing the expected value of some function [...]. This function is known as the von Neumann–Morgenstern utility function.

Let us refrain from philosophical claims about what utility functions are "actually" corresponding to in every day life. We can simply follow the practice of mathematical economists and write $u(O)$ for the real number that is associated to the outcome that a set of orders O is filled. If you want to think of it as *value created,* "combined" *user preferences,* or simply *profit:* anything is good, as long as it does not add confusion or assumptions that lead to inconsistencies. In the same spirit, we shall take one more page out of the mathematical economist's book.

[1] This may be due to the fact that I have grown up in a low-context culture, but well, let us not digress.

[2] To my dear critics: I hope that you will take the time to check that you agree with the definitions given in the first part.

[3] The elided assumption concerns "certain axioms of rational behaviour."

Discounted Utility

Assets that we have at our disposal today are more useful than those that will be in the future. This is an oddly common phenomenon. One way to capture it, is the use of

- a discount factor β and
- discounted utility $\beta^t \cdot u(O)$ for fulfilling a set of orders O after waiting for some time t.

That is all we have to know about discount factors and discounted utility.

Welfare Maximization

The last point is very simple. We want to optimize for a "global" utility. This is non-trivial in general, but will be fairly straightforward for the "idealized world" of this post: the only cost we consider is communication cost, which we want to minimize.[4]

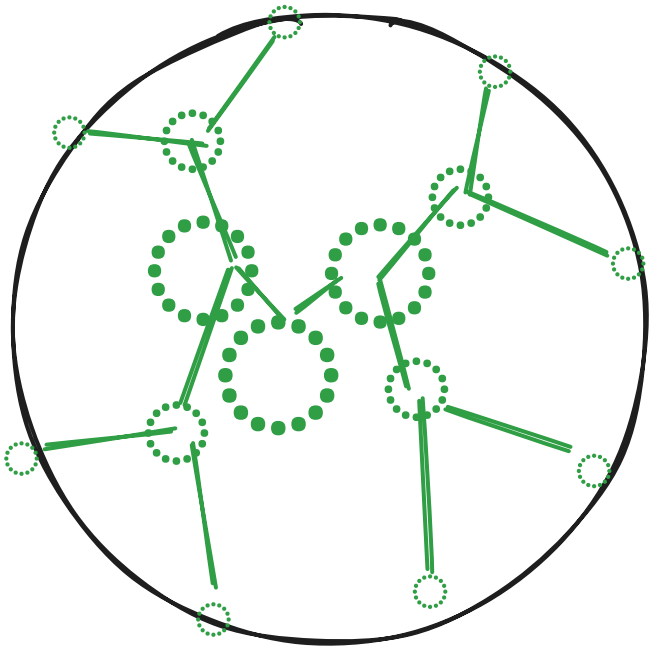

Figure 1: Our idealized one-dimensional world sphere: green operators on a black circle and their hierarchy of pools that form a binary tree.

[4]The compute cost will only add minor detail.

Back of the envelope calculations

We assume that orders may flow from any point on a circle as the black one in the figure. More precisely, we model order flow as a set of Poisson point processes equally spread around our circle of operators, rendered as green dotted small circles on the black circle.[5] We assume the activity of these order flow processes to be time-invariant so that batch auctions fill $X\%$ of all order flow in expectation where **I am missing a good value for X, but it should be around 50% or higher, but 30% may work as well.** Unfilled order flow will move along green lines towards the global pool in the centre, but has the possibility to be filled in pools of intermediate size. Let us describe this now in more detail and present some conditions that may deserve to be explored in detail.

The network structure: a complete γ-ary tree of operator pools

Suppose we have γ^k operators that can accept order flow.[6] We arrange the operators as the leaves of a complete γ-ary tree, and the inner nodes of the tree thus correspond to *operator pools* of size γ^h where h is the height of the inner node of the tree; thus, the root corresponds to the *global pool* of size γ^k and each operator could be seen as a degenerate pool of size 1.

Delay approximations and why we should care

Finding consensus among operators of a pool—be it about the next batch of intents, transaction bundles, or other types of order conglomerates—will take quadratic communication complexity, but only incur a delay that is proportional to the (average of) longest communication delays. For a first estimate, we assume operators to be spread out evenly on a circle, such that sibling leaves of the tree are neighbours on the circle.[7] As estimate for communication latency within a proper pool, we take the distance to the operator that is furthest away. Throwing in some symbols, the communication *cost* in pools is proportional to $(\gamma^h)^2 = (\gamma^{2h}) = (\gamma^2)^h$ and thus, submitting orders to a pool of double the size incurs γ^2 the communication cost, which is detrimental to welfare; in contrast, *latency* in a pool of height h is proportional to γ^h, as this is the order of magnitude of the latency (in the nice average case), but due to discounted utility, we have a factor that is exponential in the latency, namely $\beta^{(\gamma^h)}$. The latter is actually the *»raison d'être«* of why we are doing this.

Looking once more at latency: we aim to have at least one batch auction in a pool of height h before unfilled/unmatched orders are forwarded to the parent-pool at height $h+1$—early enough for the intent to be considered in the next

[5] So, we have one Poisson point process for each possible swap request, and we assume that matching swaps are of the same activity. Moreover, whenever a new swap request arises in this way, the probability to arrive a given validator is γ^{-h}, i.e., order flow is the "same" for all operators.

[6] Remember, these are back of the envelope calculations!

[7] The globe would actually give "faster" approximations.

auction in the parent-pool.[8] Thus, although $\gamma = 2$ is nice for illustrations, we may actually rather need $\gamma = 3$ in the end, but little does it matter for the big picture.

The crux: filling rate

We shall reason in terms of what we call the *filling rate* of the auctions in the single pools of the hierarchy, i.e., the expected percentage of order flow that is filled:

$$X = \frac{\sum_{O \subseteq B_n} |O| \times \hat{p}(O)}{|B_n|}$$

where B_n is the n-th intent batch, $O \subseteq B_n$ is a sub-batch of orders, and $\hat{p}(O)$ is the probability that O is the (maximally useful) set of filled orders.[9]

For a concrete simple example, we take the case of pairwise swaps, say of digitally transformed pokémon cards; each swap comes with a bid for having it filled. The matching algorithm is almost trivial **(and the reader can figure it out themselves** $;-)$**).** Note that this will match each matching pair **(and will favour smaller over bigger cycles if they are found [ooof! We need an example with a much simpler matching strategy!!!])**

Now, let us turn to the **obviously oversimplified and potentially *wrong* calculations.** Assume that we have a discount factor $\beta = 0.5$ and that batch auctions in a child pool are twice as fast as in their parent pool; **(moreover, assume that bids only differ in negligible amounts [some details need checking/experiments!!!]).** Under these conditions, we have a theoretical benefit from a hierarchical setup if less than half of the order flow has to flow to the global pool or its direct children.

The reason is that then less than half of the order flow incurs the squared discount factor, while all the matching at lower levels is benefiting from a "rooted" discount factor (or better). In concrete numbers, less than half of the orders will have the squared discount factor $\beta^2 = 0.5^2 = 0.25$ while more than half of the order flow is subject to a discount factor of $0.5^{0.5} = \sqrt{0.5} \approx 0.71$ or even closer to 1. Roughly, the hierarchy is matching faster in expectation than a single global pool. So, if we had a matching efficiency of 50%, one additional layer would already suffice and the depicted hierarchy of eight pools would already be better than a single global pool. For values below 50%, we would need a deeper tree.

[here we would have something nice to let the reader play with the numbers, using reveal.js (see, e.g., here) but we need to sync with Pedro during/after EthCC]

[8] Solving time has also to be considered, but we are only interested in situations where solving time is "almost" negligible, i.e., at least an order of magnitude below consensus time.

[9] Here, we are considering a greedy strategy. If there is no greedy strategy, we are beyond the scope of the blog post.

Assuming the filling rate as constant throughout pools is actually "wrong", as discussed in the conclusion. Before we come to that, let us discuss some **random points**.

upshot, fine print, observations, etc. [some bullets]

- Obviously, it depends on the order flow what the matching efficiency is.
- Also, it is not clear how to deal with orders that get stuck in the global pool, and we need a life time in practice.
- While there are common choices for arrival processes for modelling order flow, *filling rate* is possibly intricate to calculate.
- Note that we assume that the pools are operated by the same parties, somewhat reminiscent of sub-nets in Avalanche; the relation to Eigenlayer needs to be checked. In principle, one may have a heterogeneous setup, but then slashing and the like will be more complicated.

[**This part may go away and be replaced by actual content**]

Conclusion: the obvious non-sensical, the scale-free, ...?

In short, hierarchical mempools *could* be a better solution for solving than global pools. However, that was not the point of this post. We now identify two different reasons that can make operator pool trees "useless" (cf. distributed solving).

With the picture of the hierarchical operator pools in the circle in mind, if matching orders occur on opposing sides of the circle only, then clearly, there is no point in having a hierarchical mempool, because matching orders more often than not will travel all the way to the root pool. In this case, we only need the global pool. There is no tree.

On the other extreme, if matching orders always occur in proximity to each other, we only need sufficiently many "discrete" local pools, i.e., without connections between the pools whatsoever! At the very extreme, we have the occurrence of all order flow in a single place, e.g., due to proximity to the server of a pre-existing CEX. There is simply no point in the tree-structure.

We may now to set out to characterize in more detail a "good" category of order flow. It seems that if order flow is uniformly spread, we may have enough orders filled for each level of the hierarchy to have a *»raison d'être«*. However, we need to dig deeper, find good examples, and do some serious math. In passing, I want to mention the large body of work on self-similarity, scale-freeness, and queuing theory. Finally, although we just only scribbled some quick back of the envelope calculations, we have identified a setting where hierarchical operator pools *may* have a theoretical advantage. Time will tell if Anoma's intent machine could benefit from operator pool trees or similar hierarchical organization of intent pools.