Optimistic confirmation of transactions has been discussed as an optimization for both Bitcoin Cash (using Avalanche) and Ethereum. Theoretically, Pass and Shi (Hybrid Consensus

, 2017) showed that one cannot circumvent BFT limits and that one will need to use a purely probabilistic, non-responsive method to achieve optimistic confirmation. The current techniques for bootstrapping optimistic confirmation appear to use one of two methods:

1. Force your main chain to use a DAG structured transaction/block graph and choose an appropriate scoring function to decide if a branch should be confirmed (e.g. Inclusive Protocol, Avalanche)
2. Having two chains with one chain serving as the 'higher fork rate, faster confirmation time' chain that writes proofs / commits to a 'lower fork rate, slower confirmation time' chain (e.g. Thunderella)

The first method is strictly probabilistic in nature and has a threat model that is quite different from that of the standard UC model. The second methods have proofs of safety and liveness that are much closer to traditional UC / simulation proofs, but provide poor guarantees (e.g. Thunderella requires 75% honesty and it can be shown that you cannot do better).

Can we do better by not trying to approximate responsiveness (see the Thunderella paper for a definition) and instead focus on quantifying the maximum cartel size that we are resistant to? In voting theory and the Fourier-Walsh analysis of boolean functions, there are precise tools for quantifying the size of juntas and cartels and there are some important results like Friedgut's Junta Theorem that affect voting-based protocols. I'm not going to dive into the theory in this post (please see this draft blog post that I've written for more details and historical context), but am instead going to try to provide some intuition for how to use these tools for optimistic confirmation.

Suppose that we have a function $f : \{0,1\}^n \rightarrow \{0,1\}$

that operates on bitstrings. This function represents n

voters providing a vote and f

is the social choice function, such as majority vote Maj_n

, that decides the final outcome given their votes. The sensitivity of the f

, $S(f)$

is the number of individual bits that we can flip to change the output value of f

. On the other hand, block sensitivity of f

, $B(f)$

is the number of blocks (not necessarily consecutive bits, but sets of bits) that can be flipped to change the function's output. This measure can be thought of as a measure of minimum size of cartel that can flip the outcome of a voting function f

. The recently proved Sensitivity Conjecture (Scott Aaronson, Terence Tao) shows that $S(f) \leq B(f) \leq S(f)^2$

. This means that we can more precisely adjust our social choice function's sensitivity to cartels (quantified by the difficult to compute $B(f)$

) by optimizing our voting function f

based on it's value of $S(f)$

(which is easy to compute). In particular, if $S(f) = \Theta(\sqrt{N})$

, where N

is the number of validators, then $B(f) \in \Theta(N)$

. If we traipse through the definitions of these terms, this implies that we could optimistically confirm a transaction after getting $\sqrt{N}$

votes while knowing that we are resistant to cartel's of size $\sqrt{N}$

.

Let's look at an explicit example known asRubinstein's Function. Consider a voting function $f : \{0,1\}^{4k^2} \rightarrow \{0, 1\}$

for $k \in \mathbb{N}$

. Imagine that the input is a $2k \times 2k$

matrix of zeros and ones. We say that f(x)

is one iff there exists a row with two ones and zero otherwise. One can show by considering inputs with one 1 per row and all zeros, that S(f) = \Theta(\sqrt{n}), B(f) = \Theta(n)

, thus suggesting a quadratic split.

Given that many of the voting functions used in PoS protocols are not simply \mathsf{Maj}_n

but are instead recursive voting functions (e.g. \mathsf{Maj}^{\otimes 3}_{3^n}

) or stake-weighted linear threshold functions, there is significant room to engineer the precise voting function to allow for optimistic confirmation. In fact, the Rubinstein Function most closely resembles voting on sharded blocks and resembles @AlexAtNear's diagram in his Ethresear.ch post about NightShade.

Thanks to James Prestwich (Summa), Alexandra Berke (MIT), and Guillermo Angeris (Stanford) for helping me distill this to something easy to digest.