The current tournament has four significant flaws. In order of importance:

Problem I: Contrary models

Many of the models we receive are bad models, with a `p` significantly below 0.5. This is especially true of those with low `c` values, and is in fact incentivized by the tournament structure. As an example, consider a model staked at `c = 0`. This stake will burn if it's in the prize pool and the model fails, and will earn if it's in the prize pool and the model succeeds. However, a stake at `c = 0` will likely only be in the prize pool if most of the models above it have failed. Generally, good models will be pretty correlated and will fail at the same time. Thus, the model staked at `c = 0` is not incentivized to consistently beat the benchmark, but only to beat the benchmark *in those "hard" rounds where everyone else fails*. Thus, a model with a true `p` of 0.1 can be quite profitable -- possibly more profitable than a model with a true of 0.6 that's staked with a normal `c`.

In general, the result is that many of the models we receive are "contrary" models -- those which are designed to succeed only when others have failed. This is bad for the metamodel if it slips in, and even if we catch it and filter it out, we're still paying large amounts of the prize pool to models which are not providing value. This is a waste of money, depressing to users, and motivates people to spend time on those games rather than giving us good models.

The specific flaw here is that when one person loses, the prize pool flows down to the next person. It should never be good for one user to have another user fail.

Problem II: Optimal strategy is complex and confusing

Second, optimally participating in the tournament is hard and confusing, even if you're not pursing contrary models. Determining an optimal `c` requires a very large amount of detailed analysis. The whitepaper gives an upper bound on the maximum profitable `c`, if the payout is only in dollars. However, in practice this `c` is neither optimal nor even profitable, for several reasons, including:

- Due to the afore-mentioned correlation effects, you're more likely to burn in bad rounds and be out of the payout in good rounds.
- You want to bet the `c` that places you as the last person in the prize pool, because that gives you the best odds. Anyone who bids higher than that left money on the table.

Problem III: Monday morning staking race

Third, the optimal `c` depends a lot on what other users have bid. Thus, you gain a lot from being the last person to stake. This creates a race around 6am on Monday mornings to place the bid at the end. This isn't helpful for us, so it's just busywork/lottery, which creates a bad user experience.

Problem IV: `c` parameter conflates too many concepts

Fourth, `c` is just a number that has very little meaning. The simplest explanation of `c` is "the amount of NMR you're willing to stake to earn 1 USD and however much NMR we're giving out per dollar, which is currently 1/6." This doesn't suggest a strategy, but it's at least accurate. It incorporates the exchange rate of NMR/USD, your internal guess at `p`, and how high a `c` you think other people will bid, and how many of their stakes will fail.

We propose four changes to fix these, in reverse order of importance:

Change A: Change payout factor so that `c` is a percent consistency

Payout must be a linear factor of `s`. Currently, that factor is `1/c`. We change this to `(1-c)/c = 1/c - 1`. This change is subtle, but has significant implications. Specifically, consider the expected value equation:

```
EV = p*s*(1-c)/c - (1-p)*s
```

If `p = c`, then:

```
EV = p*s*(1-p)/p - (1-p)*s = s*(1-p) - (1-p)*s = 0
```

Thus, if your goal is to be profitable, then the worst `c` you're willing to accept should be `c = p`. This is much easier to think about, and solves problem IV.

Change B: Denominate winnings in NMR, partially paid in USD

Note that in the EV equations above, we ignored any exchange rate between USD and NMR. This is because we now denominate our prize pool in NMR, but it will be partially paid out in USD (withdrawn as ETH). There will be a fixed prize pool of X NMR, of which Y USD-worth will come as USD.

For example, consider a tournament that has a prize pool of 1000 NMR, and we determine the USD portion is 5000 USD, and the price of NMR at the time of tournament resolution is $10/NMR. Then, a total of 5000 USD and `1000 - 5000/10 = 500` NMR will be paid out. Suppose your stake had `s = 10` and `c = 0.5`, so your payout is `10*(1-0.5)/0.5 = 10 NMR`. However, you'll receive this as 50 USD and 5 NMR.

If the same tournament happened with an NMR price of $20/NMR, then a total of 5000 USD and `1000 - 5000/20 = 750` NMR would be paid out, and your stake would receive 50 USD and 7.5 NMR.

This allows us to ignore the exchange rate in the EV equations, so the math can be much simpler, and `c` just means "best guess of `p`".

This, combined with change `A`, solves Problem IV. Now, `c` is simply the consistency required of your model to make the game profitable.

Change C: Payouts for everyone happen at lowest eligible `c`

We change the payout such that everyone who is eligible for payout is evaluated as though they had bid the `c` of the last person in the prize pool. Thus, if Alice bids with `c = 0.8`, but the last person in the prize pool bids with `c = 0.5`, then her payout is the same as if she had bid `c = 0.5` (which is a better game for her). The hope is that Alice will bid the worst game that she's willing to play, but we give her the best game that she would have gotten if everyone had been up at 6am bidding against each other. This hope can't be realized without one more change, though.

Change D: Choose who's eligible for payout before scoring models

We choose who's eligible for payout *before* scoring models. Thus, on Monday after a staking tournament ends, but weeks before it resolves, we find the lowest-`c` stake such that `sum(s)` for it and all stakes above it times the payout factor (which is `1-c/c` for that lowest `c`) is still within the prize pool. Thus, who is eligible to burn/earn depends on the *potential* payout -- not the actual payout. If everyone in that group burns, there could be no payout at all. Since usually some fraction of the potential payout won't be paid (because some users will burn), we'll raise the payout pool accordingly.

This eliminates Problem I, because whether or not your model is eligible to burn/earn is *not* dependent on whether other models do poorly. Thus, you're incentivized to build models that do good always, rather than only when others do bad.

Changes C and D together solve Problem II, since there is no longer an incentive to bid with a lower `c` just so that you play a better game. Everyone who's eligible in a given week will play with the same `c`.

Changes C and D together solve Problem III, since you should just bid the worst `c` you're willing to play rather than waiting to see what other people are willing to play.

Optimal Strategy

After these changes, in general the optimal `c = p`. You shouldn't bid higher, because then your expected value is negative. You gain nothing by bidding lower since we'll give you the best `c` you could get (same as if you bid lower). You shouldn't bid lower because then you might miss out on some profitable games.

In a sentence, you should bid the worse odds you're willing to play, which is just your best guess of `p`. Then you'll play any games where you can expect to profit, and you'll bow out of games where people have bid up the odds so high that you wouldn't be profitable to play. That should be game theory optimal.