TL;DR

I believe you could use simulations, possibly with reinforcement learning, to determine what cryptoeconomic parameters to use in new token projects, or in changes to existing networks. I am wondering whether others have thought about this, and looking to start a discussion around better methodology to determine cryptoeconomic parameters for blockchain projects.

Background

Cryptoeconomic parameters, a.k.a. hyperparameters (in the AI world), are important for blockchain projects because they have a significant impact on the future distribution of tokens. Yet, today most projects seem to choose these arbitrarily. For example, an ICOing project may decide to keep 10, 20, 50, or even 90% of its tokens, contribute some of the rest to a "development fund", have a founders' reward of X%, etc. Similarly, Ethereum is currently considering reducing the block reward from 3ETH to 0.8ETH when Casper FFG goes live. In most projects, there is likely an "ideal" long-term distribution of tokens that the founding team or community would want to optimize for (ex: most tokens in the hands of users/devs vs. speculators or miners), but few projects seems to have made rigorous attempts at quantifying this, as well as the impact of potential changes on this distribution.

This post describes a high-level idea for how this could be done using reinforcement learning, and is meant to initiate a discussion around developing better methodology for hyperparameter optimization in cryptoasset & blockchain projects.

Cryptoeconomics as a RL Problem

Reinforcement learning (RL) is a sub-field of artificial intelligence, where instead of using training and testing data to teach and evaluate the model, we define agents, which can take a variety of actions over time in a specified environment. Each action produces a reward (or penalty), specific to the agent, which is quantified using a reward function. The agents must optimize their reward function over time, and will thus learn the "best" actions to take to maximise its long term reward in the environment.

This framework seems relevant to several cryptoasset and blockchain projects, as many of them have a finite set of agents (ex: users, developers, miners, speculators, hackers, etc.), which can take a finite set of actions when interacting with the project, and receive a reward (in the form of tokens or value from the project) for their actions.

Why RL?

What I believe makes cryptoassets particularily well-suited to reinforcement learning is that building a model of the environment may be simpler than in several other reinforcement learning use cases. All interactions with a blockchain happen online, and while some offline actions may be incorporated into models (ex: collusion amongst validators in a dPOS system), these seem to be the exeception and not the rule.

Also, reinforcement learning has proven very successful in finding bugs in its operating environments [1]. Most blockchain designs are fairly simple, at least compared to other types of environments such as video games or "the real world". This suggests that RL would likely be effective at finding issues with blockchain system designs.

Lastly, one of the challenges of reinforcement learning is defining clear reward functions. Fortunately, most blockchain projects have a token associated with them, which carries some value. Therefore, for several agents, the reward function could be based on accumulating tokens, with some caveats.

EDIT:

@cpfiffer pointed out in the comments that:

Another point towards RL is that we care about the equilibrium of the state (at time T) as much as we do about all the intervening steps (times t0, t1, … T) You can make a pretty strong argument that the intervening steps may describe the evolution of the system.

How

The specifics of the implementation are beyond the scope of this post, but at a high level, we can imagine building RL simulations using the following workflow:

1. Define the set of hyperparameters to optimize, the agents involved in the network, and the potential actions of each agent in the network;
2. Define reward functions for each agents taking into account all their potential interactions with the network;
3. Define an operating environment for agents that models the network;

4. Add agents to an instance of the environment with specific hyperparameters (ex: number of tokens, txn cost, mining reward, etc), have them maximise their reward function over a period of N units of time, and output the final state of the network, i.e. how many tokens does each agent have, and perhaps the amount of value they've contributed or received from the network;

5. Re-run (4) with a different set of hyperparameters for however many values of hyperparameters should be tested;

6. Perform an analysis on the results and select hyperparameters that lead to "best" final state of the network.

Issues & Caveats

Although RL could be a promising approach to determining the best hyperparameters for a network, it is far from perfect. Here are some issues associated with this approach:

1. High computational cost of simulations

2. High "barrier to entry" to build models (in other words, you need to grok both RL/AI and "the blockchain")

3. Challenges around defining environments and reward functions that mimic real-life behaviour

4. RL is non-deterministic, so two runs of the same simulation may produce different results.

For a sobering read on the current state of RL, seehttps://www.alexirpan.com/2018/02/14/rl-hard.html

Conclusion

Reinforcement learning could potentially be used to better determine hyperparameters for blockchain networks. I have outlined a high-level approach around how this could be done, as well as some of its potential pitfalls. It is important to state that reinforcement learning is simply a tool to help us acheive the goal of having optimal hyperparameters for these projects. Other approaches may be more promising, and the result of RL models should be treated as one of many inputs into the decision of which hyperparameters to use.