https://curation.network
version: 0.3
23.10.2018

A Decentralized Ranking Protocol

Introduction

Ratings have become an integral part of our everyday life. We use them for choosing gadgets, applications, and games; ratings help us plan trips and careers. Even your Facebook feed and Google search results are unique ratings.
They are indispensable when we are trying to make the best choice among many available options but are lacking data, experience or knowledge. For App Store, Steam, Booking, Amazon, Glassdoor, AngelList, Yelp and many other companies ratings are an important service component.

That said, all existing ratings have vulnerabilities related to their centralized nature, such as susceptibility to censorship, manipulation for commercial purposes, bias, and vague understanding of their formation rules. The impact of ratings on purchasing decisions is hard to overestimate: they have become a tool for influencing the market and manipulating consumer consciousness. For now, the only way to resolve these problems is to compare conflicting data from different sources (ratings).

We aim to resolve these issues with the help of a decentralized rating system based on blockchain technology. Our system relies on community experience such as Token Curated Registries, Curation and Prediction Markets, and our own dynamic ranking models that further develop these concepts.

The main value of our solution is that it allows us to compile objective, independent, censorship-resistant and manipulation-proof ratings. Also, our algorithms are one of the necessary basic blocks for building DAO models.

We believe that the solution to the above-stated problem is economic algorithms built on game theory basis. They involve gains for performing rating-useful actions and losses - for the harmful ones. Action usefulness is defined by an increase in consumer value of the curated rating. Its information enrichment and status reduction most accurately reflect the community opinion.

Economic games have been modelled and described in detail by modern economists, and smart contracts are an excellent tool for practical implementation of these models. The token concept allows creating automated economies based on these models, with full transparency and excellent data quality. We believe that decentralized rating protocols will

become one of the basic blocks of such economies and maximize the benefits of various registers, ratings, and other supervised data.

Certain conditions should be met to make decentralized ratings possible. There should be many independent participants, economically incentivized in improving the quality of ratings. They form specific rating metrics. This, in turn, infers the need to create game theory-based protocols that make actions of an honest curator profitable and unprofitable for a cheater. Any attempts to manipulate a rating or occupy the registry, especially through technological superiority, greater computing power or advanced automated tools, will be penalized.

Our work will result in a system that effectively solves the problem of decentralized rating curation. It will allow everyone to use the existing ratings managed by expert curators or launch their own registries, define the rules and principles of work, value for the target audience, and create their own token economy.

In this case, registries should be based upon subjective assessments of curators and not lose humanity, since the ultimate goal is ranking, relevant to the subjective opinion of the most influential curators. Moreover, we allow and even welcome the opportunity to influence the rating in a limited, transparent and temporary manner by consciously redistributing our tokens in favor of other holders, and thus drawing community attention to certain elements of the rating.

System requirements

The system should encourage honest curators to thoroughly monitor the quality of the registry, rewarding them for high-quality rankings while making other low quality strategies unprofitable. It is unfettered human opinions that matter the most, so our goal is to encourage curators to apply their expertise, share vision, cooperate, and create tools that protect the registry and support its high quality.

Gaming models of ranking protocols should not only favour good actions and penalize the bad ones, but also be immune to various mechanics aiming to sabotage the protocols through inaction or using automatic tools. The model must contain data that does not allow it to fall into stagnation or uncontrolled growth. Also, the model should be resilient to a so-called “cold start” and stimulate curation growth. The system is to develop rapidly at the beginning, with increasing level of engagement, but at the same time allow for fewer radical changes in rankings.

The system should be curator-friendly, taking into account different behavior models: regular, episodic, or even random supervision. Curators need to have relevant data to make well-thought decisions and avoid random and systematic errors. Also, the system should be simple, cross-usable and available for a wide range of users.

Token curation models should take into account the specifics of the subject area. They can limit the allowed rate of rating changes, use nonlinear scenarios to calculate metrics, require validation of complex operations performed by several participants, and use other methods to protect supervised data from unwanted changes, in particular, overly fast or voluminous ones.

Formulating the objective

Our objective is to develop tools for creating and maintaining a ranked list that can be managed by a potentially unlimited number of curators. Their opinion value is determined by their personal special token balances. For example, in previous system versions, the Curation Network served a CRN token.

In this paper, we will omit token macroeconomics, i.e. its exchange rate to other tokens, such as ETH, and its value as a payment asset. These factors are extremely important, but our goal is to create a universal decentralized rating protocol. At the same time, following basic economic principles, we are confident that the value of any token with data supervision functions will reflect its value for the community.

In curating systems, a token is primarily an asset in an economic game with zero or non-zero amount; it is advantageous to improve the quality of data rating, and it is not profitable to spoil and spam it. We shall bear in mind that we are concentrating on creating a list of honest people, rather than metrics or engineering, which means that the algorithm should reflect the natural behavior of people in society. That is, the algorithm must model the limitations and assumptions that people are used to, evaluating various facts around us in real life.

The solution to building a ranked list is already present in several protocol concepts, such as DPoS consensus (voting for delegates) or a TCR model modification called "graded TCR". In these protocols, the curator freezes (stakes) a certain number of tokens, voting for the selected object. The position of an object in the rating is determined by the sum of votes cast for the object (frozen tokens). With certain limitations, this scheme is operable enough to build small lists in the laboratory, it is simple and easy to use.

Unfortunately, these schemes do not solve the problems that arise when a rating is supervised by an unlimited community of real people with various interests. Before we proceed to the description of our model, let's outline the main issues and types of attacks that the protocols are subject to.

“Big Stake” problem

This involves voting with a bigger amount of tokens/votes than other participants have. Potentially, the owner of a large number of tokens can quickly transfer his tokens/votes to his object of interest, dramatically lowering and raising his position in the rating. Even if we do

not take into account malicious use of a large token stake, such vivid manipulations with a ranked list are unlikely to increase its value to users.

In case of malicious activities, the owner of a large number of tokens can easily make a list inoperable, for example, adding and discarding random objects. To handle this problem, the curation algorithm should make such activity unprofitable or at least as risky as possible.

"Sybil stakes" problem

The so-called Sybil attacks are possible in case of well-coordinated actions of a large number of account bots. Even if our algorithm coped with the previous problem of "whales" (for example, by setting up restrictions on the voting stake size or penalties for mass operations), this is still not enough to protect against a large stake distributed among a number of accounts acting in collusion.

This is one of the most serious problems of decentralized algorithms which is often underestimated, because for now such attacks are unprofitable for attackers, and blockchain applications cannot yet offer the attacker serious rewards for mass automated actions. That said, as soon as the ranking is valuable, sybil attacks may become extremely relevant (for instance, Steemit and Golos cases).

Constants-restrictions imposed on operations are not a good solution to protect against such attacks, so the model itself should make actions of Sybil accounts unprofitable.

The "Always winning stake" problem

This problem is common for all systems where participants operate with token balances for profit. Any system that guarantees to reward participants for some action is subject to this attack.

If the system has an action one can always get a reward for, then it is possible to create a bot that strictly performs this action and receives a small but guaranteed reward for it. In this case, the system gets biased in favor of income-generating actions, and becomes useless.

To protect against this type of attack, the model must implement a certain zero-sum economic game to make massively coordinated actions unprofitable, and if they occur, their effect should be equivalent to simple random guessing.

"Lazy stake" problem

That is inaction of a curator that may happen at the beginning of a new rating formation involving a lot of objects and few curators. In this case, the algorithm should motivate curators to build a balanced rating.

Considering that curators are ordinary people and it is desirable to engage them into activities gently and without coercion, this issue is also extremely difficult to solve. Human activity is irregular, unevenly distributed over time and rather unpredictable.

Consequently, the model should be able to accelerate processes while attenuating the level of curator activity and vice versa.

"Init stake" problem

This is a "cold start" problem. Curation of some registry involves the initial token distribution which can be either random or, on the contrary, too specific. Depending on the model, token distribution may be between a limited group of experts or a huge number of random curators.

The model should work effectively with any initial numbers of participants, stimulating curation at the beginning and gradually requiring less and less active actions from the curators of the registry as it grows up.

The problem of simplicity and accessibility of presentation

A system for users must be understandable, predictable and correctly display the feedback. It is essential that curator work results and the system general state can be shown when necessary. In case of dynamic systems this task becomes rather complicated, since they should be able to display several data series and potential outcomes of curator actions.

Another challenge is to create user-friendly UI elements including curation management options.

CRank Protocol (Algorithm)

We believe that current models operating on absolute balance values and allowing to vote for individual objects by simply freezing the required number of tokens (gTCR, DPoS) can hardly resist the above-described problems. In particular, they are applicable to curation based on subjective evaluation rather than on objective data. We offer a more comprehensive solution enabling curators to interact with the registry and minimizing consequences of the issues described above.

Our approach is an attempt to move from schemes using absolute values to ones that take dynamic behavior into account. In other words, basic system parameters are variable and depend both on global parameters and processes occurring in the system.

In nature, there are many systems that are regulated by dynamic parameters: from intracellular reactions to biocenosis. Similarly, dynamic complexity of the Bitcoin network is defined by overall network speed. In such systems, we can observe dynamic equilibrium, ensured by coordinated actions of many participants.

We assume that this approach will allow us to create viable systems, resistant to external influence and always striving for balance and self-regulation.

Feature 1: Impulse model

The first important feature of the algorithm is a dynamic model. This means that any curator action in the rating takes some time, i.e. it does not instantly change a vertical coordinate of the object (as in simple stake vote) but gives the object some momentum, after which it is moving for a while, changing its position in the registry.

A good physical analogy is a body immersed in liquid that, after receiving some impulse, floats or sinks, taking different positions on the depth scale. To move the object, the curator uses a certain number of tokens, setting the impulse strength and direction. The total impulse is determined in the course of curators' voting and afterwards is applied to the object to move it in the right direction. Time dependence of coordinates can be nonlinear or limited.

Thus, the curator’s actions become more profitable in more or less “standard” voting stake, and unprofitable in case of too large or too small stakes. This can be ensured by various nonlinear dependencies of pulses.

Such an approach will help our algorithm better resist harmful effects and become more curator-friendly and understandable. Also, having proper selection of system parameters and feedback, the registry will automatically strive for the optimal change rate, stimulating or slowing down the movement of objects when needed.

Feature 2: Feedback

The second feature is inverse relationship between the nature of object movements and system parameters, such as awards, fines, the cost of imparting impulses to objects, restrictions on the number of actions in the system and other parameters affecting the system equilibrium. Basing on simulation results and actual testing, we plan to refine the model feedback in order to:

1. Compensate for "init stake": increase the change rate in the initial registry state, when objects are not yet ranged and higher returns are required; or slow down the process, when the registry already has a large number of active curators with a small stake.

2. Make up for the "big stake" problem by increasing the cost of voting, the size of commissions and other parameters that make mass operations with big stake unprofitable. But only after the registry is initially “configured” since, at the initial stages, a small number of curators with huge token balances and drastic changes in the registry are normal.

3. Maintain a balance between the risk and return for curators to mitigate the "sybil stake" problem as much as possible; keep automated bots, as their random behavior minimally harms the registry.

Basic algorithm

Glossary

Curator — an address with a certain token balance, using which transactions are performed in a TCRank contract.

Voting — a procedure of voting for some action (such as inclusion of a new object in the register (application) or sending an impulse to an object). In earlier versions, we use a simple "commit-reveal" voting, so that the curators could not predict the final object impulse till the end of the voting and change their decision.

In later versions, we plan to switch to *"zero-knowledge"* variants of (completely) anonymous voting and pseudo-random generation of curation awards and fines in order to minimize the risk of "Sybil stake" attacks.

Impulse — a mathematical value that determines the intensity and direction along which curators move an object from its initial position in the registry. At the moment, voting is over and the resulting impulse, its time and intensity have been determined. From now on, you can calculate the position of the object at any time, according to the equation of motion.

TCR (Token Curated Registry) — common name for curation algorithms involving voting via balances of a certain token.

General scheme

In general, curation of a certain list of objects consists of two subsystems:

1) Content management system - for managing object metadata. The purpose is to provide the registry with valid and honest data about objects, exclude malicious content, change object data, correct errors and improve quality. This system should enable curators to add, delete and correct object metadata and ensure high-quality moderation of each operation with an object in order to prevent errors and malicious usage.

2) Ranking system - for ranking objects in the registry. The purpose is to build a rating of objects that are useful for the end user from the curators' point of view. Curators must be able to control the position of objects in the registry, moving objects up and down in the ranking. The system should make it as difficult as possible to reduce the quality of ranking and value to users.

Content management system

Using the standard TCR algorithm, curators include new objects and eliminate the old ones. It is required that a certain number of tokens be frozen. To unfreeze them, we can only delete the object from the registry, i.e. tokens are frozen till the item is on the list.

To include (or eliminate) the object in the registry, the curator sends an "application" to the contract (smart contract), that any curator can challenge by running a voting protocol. After voting is completed, the object is added to or deleted from the registry (via simple majority). The winners share the stakes of the losers, including the initiator of the protocol. This part should be implemented using TCR (Token Curated Registries) scheme.

Ranking system

Interaction between the curators and the ranking system consists of the following steps:

1. The curator chooses an object that deserves a higher or lower position in the ratings than the current one.

2. The curator assesses his token balance and the risk of disagreement with other curators, then decides whether to start voting or to participate in one of the current ones, and determines the rate size in tokens.

3. The curator starts voting or joins the current one, fixing the cryptographic hash in the parameters of the input impulse of the transaction. (commit phase)

4. Other curators aiming to join the vote also post their commits using encrypted impulse parameters.

5. Upon the commitment phase completion, curators open up their impulses and transfer the corresponding amount of tokens (reveal phase).

6. After the "reveal phase", any curator can complete the voting and start moving the object within the registry. Then stakes and commissions of the winners are gradually returning to their balances while the chosen object is moving up or down the ratings.

Imparting momentum to the object

This procedure is rather simple. When a curator wants to raise or lower an object in the registry, he launches a voting protocol consisting of several phases, similar to the traditional “commit-reveal” scheme.

Phase 1: Commit

A curator starts a vote on a given object, introducing the first vote with a desired impulse. He sends in token number data in a hashed form, enabling him to vote, and the impulse direction.

Then we denote a voting stake as ***Si***, and keep in mind that this is a signed number since the impulse can be positive or negative, depending on the desired movement direction of the object. Other curators also join the vote by sending the encrypted stake and impulse data.

At this stage, as the stake size is unknown, each curator is charged a fixed part of the commission (for details, please go to the section with a numerical algorithm description). A curator loses the commission in case the protocol fails.

Phase 2: Reveal

During the reveal phase, each curator makes a transaction that reveals impulse direction and value and confirms a previously published commit.

In case a curator fails to reveal his stake (bid) or there aren’t enough tokens to be frozen at his address (balance), he loses the previous part of the commission (a fixed one), and his momentum (vote) is not taken into account. When a voting stake is known, a curator pays the second (dynamic) part of the commission that depends on the stake size. The size of this commission is non-linearly related to the voting stake amount so as not to give a vote with too large or too small stake.

The algorithm for calculating the dynamic part of the commission is also described in detail in the next section.

Phase 3: Movement

After the reveal phase, any participant can send a transaction that closes up (completes) the voting process. At this moment, all open impulses are summed up, the final impulse and the time of movement start are calculated and recorded. The phase of movement begins.

Speed and position of the object can change every second depending on the applied impulse. Speed is measured in “tokens per second,” and position in the registry is determined by the object’s coordinate at the required moment of time and depends on the given equation of motion. Thus, impulse affects the final object coordinate (its final position). The equation of motion is determined by a special function defined in the next section.

The speed of the object also depends on the return rate of the invested funds (voting stakes in tokens).

Numerical description of the algorithm

The algorithm has several basic constants and functions that help to calculate the current position of the object in the rating, rewards and penalties for curators, rating “inertia” and the funds return rate. Note that the rating of objects (a vertical coordinate) is measured in tokens.

Title	Function/constant	Units	Description
V_u	constant	tokens / sec	unstake speed, i.e.object speed in the ranking. Determines the return rate of the voting stake with a reward or fine.
K_{fixed}	constant	-	coefficient to calculate a *fixed* part of the commission in the *commit* phase
$fixedFee(S_{avg},\ R_i)$	function	tokens	a fixed commission amount paid by the curator in the *commit* phase. Depends on the object rank (*Ri*) and the average size of a voting stake *(Savg)*
$K_{overstake}$	constant	-	coefficient to determine the stake size after which the quadratic part of the commission will be used
$K_{dynamic}$	constant	-	coefficient to calculate the linear dynamic commission part paid by the curator in the reveal phase

$dynamicFee(S_i)$	function	tokens	the amount of commission, defined by the size of the voting stake, paid by the curator in the reveal phase.
$movementFunc(S_{result},\ \Delta t)$	function	tokens	the object motion function in the rating, where *Sresult* is the size of the resulting impulse, and *t* is the time interval. Determines how the object will move based on the vote results held at some point of time. Get back Δr : change of the object's rank for Δt

Values and functions used in version 1

Motion function ($movementFunc(S_{result},\ \Delta t)$) :

In the first implementation, we use a linear function, which performs the motion of an object by rating it at a constant speed, until the object moves at some distance determined by voting. Then the movement stops and the object stops moving, the movement function can be defined as:

$$\Delta r = \Delta t * V_u,\ if\ \Delta t < S_{result} / V_u$$
$$\Delta r = S_{result},\ if\ \Delta t \geq S_{result} / V_u$$

where

$V_u = 0.01 * 10^{18}$: deposit return rate, tokens/sec

Δt : time interval, sec,

$S_{result} = \sum_{i=1}^{n} S_i$: resulting stake percentage

Fixed commision ($fixedFee(S_{avg},\ R_i)$):

A voter is charged a fixed commission part in the commit phase. At this moment, we do not know what stake he voted for, but it is necessary to ensure that a curator will not open his vote in the reveal phase.This commission part protects against "Sybil stake" attacks. In addition, the following aspects should be considered:

- The amount of commission is related to the average size of a voting stake throughout the registry. As large stakes increase operation costs, curators are encouraged to work with smaller stakes, balancing the system and minimizing the risks of the “big stake” problem.

- As operations with objects of a higher rank are more expensive, the size of a fixed commission part also depends on a relative object position in the rating. This reflects the value of top ranking positions and prevents their manipulation.

Therefore, a fixed commission amount in the first implementation is calculated as follows:

$$F_i = S_{avg} * K_{fixed} * R_i / R_{max}, \text{ if } R_i \leq R_{max}$$
$$F_i = S_{avg} * K_{fixed}, \text{ if } R_i > R_{max}$$

where

K_{fixed} = 0.1 : a coefficient for calculating a fixed-part commission

S_{avg} : average size of a voting stake

R_{max} : maximum rank in the registry

R_i : rank of the current object

Dynamic commission ($dynamicFee(S_i)$):

During the reveal phase a voter is charged a dynamic commission part. At this moment, his voting stake Si becomes known, and the commission serves to limit the curator’s impact on the registry. An average voting stake size is set; to calculate the actions that exceed it, an exponential scheme is used. Voting with a large stake becomes extremely disadvantageous.

The dynamic commission part in the first implementation consists of two sub-parts: a linear one and a quadratic one. Each is calculated as follows:

$$F_{lin} = S_i * K_{dynamic}, \text{ if } S_i \leq S_{avg}$$
$$F_{lin} = S_{avg} * K_{dynamic} + F_{sqr}, \text{ if } S_i > S_{avg}$$

where F_{sqr} - a quadratic part, calculated as follows:

$$F_{sqr} = ((S_i - S_{avg}) * \frac{\sqrt{S_{total} - S_i}}{K_{overstake} * S_{avg}})^2$$

where

$S_{total} = 1\ 000\ 000\ * 10^{18}$: total token supply (max stake size)

S_{avg} : average voting stake size

$K_{overstake} = 2500$: a coefficient to determine a stake size for a quadratic commission part

For the nonlinear part of the commission, we use a quadratic dependence of the commission size on the voting stake size.

Ranking algorithm development

The above-mentioned ratios are not final. To work with diverse registers, all the constants and functions should undergo mathematical modelling and extensive testing. We plan to formalize the algorithm as much as possible and provide a library allowing to model the CRank algorithm, evaluate various configurations and try different combinations of parameters.

We believe that both theoretical and practical parts of the system are important. First, we plan to launch a test version of the product and then refine a mathematical model.

The inertial mechanics of moving objects and returning a voting stake in our algorithm can be more complicated than a simple linear movement. It may include movement with acceleration, interaction between objects, constant medium drift (constant movement of objects up or down), or feedback from curators.

System tokens may undergo further refinement: no massive or too frequent operations to transfer large token amounts between accounts. Such restrictions can also diminish harmful use of the CRank registry.

We plan to develop a voting procedure in a way that conceals the size of voting stakes and makes the behavioral analysis more complicated. In this case, a simple “commit-reveal” scheme can be replaced with a more appropriate anonymous one, including pseudo-randomly generated commission amounts. In an ideal situation, the system will operate completely anonymous token balances, with all awards and penalties charged in a zero-knowledge mode.

Curation Network planning

Our task is not only to develop an algorithm but a working system that proves its efficiency and value. We chose blockchain projects for initial curation. We assume that interaction with DApps, crypto wallets and blockchains may interest our target audience - people keen on decentralized networks and crypto enthusiasts.

It is essential that the target audience possesses the tools to work with public blockchains and sufficient experience to interact with tokens (ERC-20 in particular). Current underdevelopment of client software and inability to use it is a serious obstacle to develop decentralized curation algorithms, and we hope that our curators will significantly improve the quality of decentralized software by their actions and honest feedback.

Another implementation feature is automated tools for working with CRank. We have studied a large number of bot services (product reviews, content portals, reputation systems) and came to the conclusion that the system must initially be resistant to massively coordinated actions.

To do this, you must make use of automatic tools available to each curator. The result should be extremely effective semi-automatic tools for managing critical registries that are at least resistant to attacks with technical superiority.

Curators can operate a complex automated registry tool to perform massive operations, for example:

1. Manage all phases of voting and ensure automatic collection of various prizes and returned stakes in a few simple steps.
2. Some strategies run automatically but curators still review the data that require human involvement to make a decision.
3. Collect and analyze various metrics that help achieve higher effectiveness.
4. Calculate optimal risk-free voting parameters and model trends.
5. Initially form the registry, placing the objects in the original order.

Paid access to complex automatic tools can be the basis for project monetization and team support.

Technical roadmap

1. Implement the first version of the CRank smart contract, unit tests and bench tests.
2. Initialize and publish the first version of the CRank contract to rank blockchain projects in the testnet.
3. Launch a DApp catalogue to display the most relevant data for curators and see the projects ranked by CRank.
4. Provide curators with UI for manual rating curation.
5. Run a full cycle of “development-testing-calculation-feedback” in the testnet.
6. Launch a Content management system to handle DApps metadata based on TCR algorithm.
7. Choose beta testers. All active curators might be awarded tokens after the mainnet launch.
8. Start curation process and metrics collection to assess overall quality of the model.

9. Analyze CRank performance in the testnet, refine and write the code simulating the work of registries with different motion functions and calculate commissions.
10. Launch a full development-test-launch-feedback cycle in the mainnet.
11. Economically motivate a wide range of curators to participate in algorithm testing.
12. Compare algorithm performance in different subject areas and create several registries managed by experts and communities.
13. Record the results in the CRank algorithm version 1.0.
14. Initiate studies on transition to zero-knowledge voting options and pseudo-random generation of awards and penalties, up to a completely anonymous voting token to further counteract automatic strategies.