Cardstack Reward Model: Proportional Attribution and Allocation Model in a Dependency Network with Intervention in the Local Scope (PAAM)

Justin Thong and Chris Tse

***Abstract*— This paper discusses a broad framework that enables the proportional distribution of rewards to nodes within a network (global scope), which is novelly defined as a union of trees (local scope). In contrast to the usual flat giveaway to nodes, we distribute wealth through trees of the network, by starting at particular nodes and propagating the rewards down to the other nodes. In this process, which we call the wealth trickle, the amount of wealth distributed to the nodes globally is proportional to the importance of each node within the global network. We establish evaluation metrics to formalise this notion of the definition of proportionality – the fairness metric. Through the observation of networks that naturally possess the scale-free property, we introduce a transformation meant to counter-balance the unequal distribution of wealth, using a global reward. We briefly consider two methods of combating the adversarial gaming of parties within the network, by involving the participants of the network in the government process of distributing wealth. Finally, we introduce an application of the framework in the blockchain space by describing an off-chain consensus protocol, Proof-of-Analytics. This paper is intended to provide a plausible solution for a problem that is part of ongoing research.**

I. INTRODUCTION

The utility of software packages is ubiquitous in our modern world, as evidenced by applications for numerical computing, server management and web development. It is an undeniable fact that the software being built these days is becoming more and more integrated. A particular web application can have hundreds of core dependencies, without which the application would not exist. This culture of reusing software is especially widespread in open-source communities, such as Github, where software code is easily accessed and shared. Our objective in building this framework is to ensure that, if a particular software receives a reward, this reward is passed on to its dependencies, the dependencies of its dependencies, and so forth. The idea is to reward software that has contributed to the success of revenue-generating software and that would not be rewarded otherwise.

What we need to ask ourselves now is this: "How can we propagate these rewards proportionally, allocating them to software dependencies based on their level of contribution?" and "Under which circumstances are these distributions fair?". These two open-ended questions have been considered by academic literature from different perspectives.

Fair division approaches these questions through game-theoretic methods. A familiar example of fair division is cake-cutting. Fair division stems from the idea of the *Subjective Theory of Value*, according to which the value of a particular resource or set of resources is subjective, depending on the players [6]. Since each player places a different value on the resource, an optimal equilibrium can be attained – fulfilling various definitions of fairness, e.g. being proportionally fair, envy-free, or pareto-optimal.

A sub-body of literature that is more current is the study of "importance" in large networks. Search engines have attempted to organise a large graph of web pages by ranking them, using content and link-based analysis. *Term Frequency Inverse Document Frequency*(tf-idf) and the vector space model are used to encode a vector space based on the textual content of documents [5]. Because real vector spaces have natural notions of distances, documents can be ranked according to their distances from each other or their distances from a search query. Link-based analyses in networks are explicitly used to determine how "important" or "central" a node is within a network. *Google PageRank* [8] is one of the most robust centrality measures that signal the "importance" of the web page, based on the number of web pages that link to it and the "importance" of those web pages.

The fields of telecommunication and wireless networks use fairness metrics for the scheduling algorithms for packet flows within the network as a means of preventing congestion of data flow. For example, proportional fairness in queueing algorithms means that the rate a particular channel receives is proportional to the cost of delivering a packet; therefore, cheaper packets are transferred first. This is also known as *Weighted Fair Queueing.* In economics, many indexes or measures of the inequality of wealth distribution have been developed, such as the *Gini Coefficient*, the *Theil Index*, and the *Diversity Index.* Each of these indexes is based on some statistical justification that represents the assumption of an equal distribution of wealth, without concern for identifying players with a particular share.

Although much literature appears to be hoping that a solution to the problem has already been devised, the problem persists in its unique nature – the duality of a network and many trees within the network, whereby different trees receive different rewards. Furthermore, wealth distribution in reputation-based systems, such as PageRank, is very unequal by nature. Reward allocations based on some value judgment of ”importance” systemically lead to particular reward collectors receiving disproportionate amounts of rewards; this deviates from our intuitive judgment of equality. Therefore, we propose a two-step reward system that maintains the proportionality of wealth based on ”importance” and adjusts inequalities through a global reward. This resembles the common nature of a mixed economy, where people are rewarded in a competitive environment based on their ”importance” to a sub-economy, while government interventions are used to aid people who lack opportunities.

II. Problem Statement

In this section, we formalise the problem. Since the reward is distributed in two different ways, we define actions for particular rewards in two different scopes: a local scope and a global scope. A scope is simply a space where action occurs. The conventions we will use for the rest of the paper distinguish between the actions of each scope.

A. Graph Tree (Local Scope)

The *local* scope is a directed tree graph, $\mathcal{G}$ structure. We use the word *local* to emphasize the context, meaning that the tree is a small part of a larger network. The graph has directed edges, which indicate that a node ”is dependent on” another node, similar to package dependencies in software. The equation below expresses the i-th tree in terms of its set of nodes and edges.

$$\mathcal{G}^i = (V(\mathcal{G}^i), E(\mathcal{G}^i)) \tag{1}$$

In a local scope, the root node becomes the source of incoming wealth, where the reward arrives before it is distributed to the dependencies. Each node carries a set of attributes, which are:

1) The proportion that determines the reward to be received by the node itself, τ. We can possibly make τ a function of the node located in a tree. For simplicity, this parameter is fixed for all nodes in the local scope (and the global scope).
2) The proportion that determines the reward to be passed on to and received by the first-level dependencies: $\mathbf{p}$, also known as a *branch attribution*. For now, we assume that this is the final *branch attribution*. In the context of filtering, these proportions may be referred to differently – section V-B

It is possible for a node to be a member of more than two different trees; therefore, it is essential to know which local scope is being referred to, because a node may have more than two different sets of node attributes.

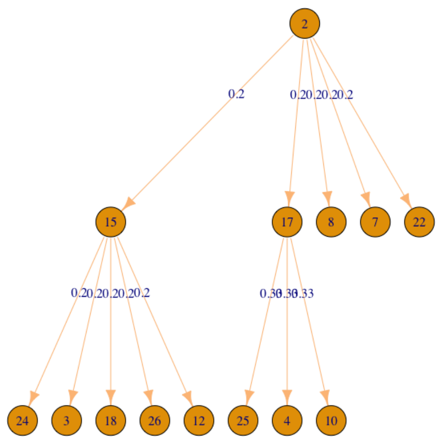

Fig. 1. Local scope(a tree) with random node ID’s. Node 2 is the root node. The weights placed on the edges are the branch attributions – see section II-D

B. Network (Global Scope)

A network, ξ, is a union of graph trees and refers to the *global* scope of our space. Although the operation of union is well defined for edges and vertices, it is not for the node attributes. The node attributes are excluded, because there is no purpose in preserving the information at the global scope. The network becomes a simple, directed, and unweighted graph. We make the assumption that no two nodes can have edges pointing to each other, thereby avoiding the formation of circular dependencies

$$\xi = \bigcup_{i=1}^{n} \mathcal{G}_t^i \tag{2}$$

$$\xi = (V(\xi), E(\xi)) \tag{3}$$

C. Reward

Our model distributes rewards. Here, we define the type of reward actions that take place. Note: The words ”wealth” and ”reward” are used interchangeably. There are two types of rewards, and their difference lies in the scope.

1) **Local Reward:** The reward given to a node in the local scope, as a result of incoming wealth to the root node. Because rewards are passed down each branch of a tree, more than one node in the local scope receives one. And the reward received by a particular node is determined by the node attributes of its parent – see section II-E
2) **Global Reward:** The reward given to every node in the network as a flat giveaway. The global reward acts as a wealth-adjusting policy, which resolves severe imbalances in the distribution of wealth. This is also known as the ”diversity reward”.

The rewards are given in sequence – beginning with all the local rewards, followed by the global reward. Both rewards will alter the total distribution of wealth within the network.

D. Type of Attribution

An attribution is a proportion determining how much a node deserves, given that some wealth is to be divided amongst a set of nodes. For simplicity, we refer to an attribution as a vector of proportions and not only a single proportion, i.e. all components of an attribution, summed up, equal 1. There are two types of attribution: The *global attribution* and the *branch attribution.*

1) **Global attribution:** The attribution to the network, which determines the global reward of each node in the network, by multiplying the wealth given to the network with each node's attribution.
2) **Branch attribution:** The attribution that determines the reward to be passed down to and received by the first-level dependencies: $\mathbf{p}$. The word *branch* refers to a single branch of a tree. Because one attribution targets a single branch of nodes, there are several *branch attributions* within a local scope.

E. Wealth Trickle

In this section, we consider the fundamental building blocks of the network – the tree structure and the way in which rewards are propagated from the root node to its branches. We call this mechanism the *wealth trickle*; this is how rewards are distributed in the local scope. Essentially, wealth obtained by the root node "trickles" down the tree, whereby each participating node obtains a portion of the wealth coming from the root node, after its ancestors have received their share. If the remaining wealth passes a branch, it is divided according to the branch attribution encoded by the branch's parent.

The notation to locate a node in the i-th tree needs to specify the j-th layer, the k-th branch in the j-th layer, and the l-th node in the k-th branch of the j-th layer, i.e. x^{ijkl}. If x^{ijkl} is specified, one unique path exists to connect it with the tree's root; the length of the path is its layer index, j. Each index begins at 0. In the global scope, the labels for the trees are not unique; one node may have more than one index label. Yet, there is only one instance of the node acting as a root node. That being said, the labels are important for all rewards in the local scope.

In the wealth distribution system, the root node of the tree, x^{i000}, receives a wealth of W^{i000}. Consider an arbitrary node x^{ijkl}. x^{ijkl} takes τ^{ijkl} of the wealth, W^{ijkl} being what it receives. After node x^{ijkl} receives W^{ijkl}, it passes down $(1-\tau^{ijkl})W^{ijkl}$ to its dependents. In particular, node x^{ij+1rs} receives $W^{ij+1rs} = (1-\tau^{ijkl})p^{ij+1rs}W^{ijkl}$, where r and s are valid indexes corresponding to layer j. p^{ij+1rs} is a component of the branch attribution $\mathbf{p^{ijk}}$ and part of the node attribute of x^{ijkl}. More generally, the wealth retained by x^{ijkl} as part of the local reward can be written as:

$$W^{ijkl} = W^{i000}\tau^{ijkl} \prod_{qrs|i\in Path(ijkl)} (1-\tau^{iqrs})p^{iqrs} \tag{4}$$

$Path()$ denotes the path from x^{i000} to x^{ijkl}. The formula above should not be over-complicated by the notation, as it is simple arithmetic whilst passing wealth down every node of a tree. There exists a point at which the wealth being passed down equals a very small number ϵ and is negligible; therefore, we should halt the transversal of the tree when this condition is fulfilled. This is useful for computation concerns as well.

$$\sum_{rs|ij} W^{ijrs} < \epsilon \tag{5}$$

It is observed that the wealth depletes, as it is passed further and further down the tree. Depending on the values of $\mathbf{p}$s for a particular layer, more branches lead to a higher rate of depletion of wealth. Almost instantly, there exist many parallels to a probability tree with discrete outcomes, since W^{i000} and $\tau^{ijkl}\prod_{qrs|i\in Path(ijkl)}(1-\tau^{iqrs})$ are simply constants. Therefore, the wealth x^{ijkl} obtains simply consists of these two constants, multiplied by the "probability" of the event of node x^{ijkl} occurring. The computation of "probabilities" can be easily executed by multiplying a weighted adjacency matrix by itself. Since we are multiplying the trickling wealth by a proportion, the wealth trickle never truly depletes; there exists a residual wealth, r^i. We have two options: We can either send r^i to the global reward pool or recycle r^i and use the wealth trickle again. This is a design decision, depending on whether equality or fairness is preferred. If the former is chosen, more wealth will be allocated with the aim of reducing wealth inequality. If the latter is chosen, more wealth will be allocated to nodes located higher in the tree. We treat this section as a formalism of the framework and it is used for further reference.

F. Evaluation Metrics

It is incumbent on the writer to establish a metric our model is judged by, or at least clarify what the model attempts to achieve in quantitative terms: There are essentially two metrics that concern us – one pertaining to a local reward, the other pertaining to the global reward after all the local rewards have been issued.

Several "fairness" measures exist to measure "fairness". The double quotes around the word "fairness" are intended to emphasize not only that the definition is subjective, but that there are different formal definitions in the body of literature. We use the definition of fairness proposed by Jain, according to which a fair attribution occurs, if each node obtains the amount of wealth that corresponds to the cost it paid, relative to other nodes [1]. We vary the interpretation to mean that a node's "importance" is inversely proportional to the cost, i.e.

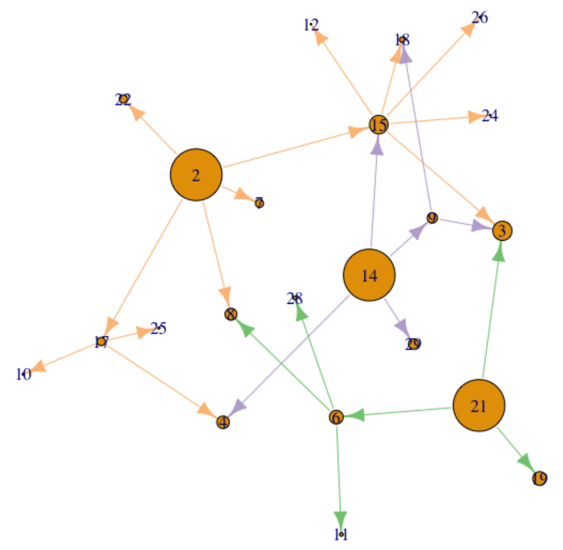

Fig. 2. Global Scope(Union of Trees) with random node ID's, inclusive of tree in figure 1. Size of each nodes correspond to the cumulative wealth collected by it

if each node obtains its entitlement, the Fairness Index will be 1. In our framework, we assume that a node's entitlement is its "importance" score, which reflects how central a node is to a network – see section III-A. Below is the Fairness Index, F, introduced by Jain [1].

$$F = \frac{[\sum_{i=1}^{n}(\frac{1}{s^i})W^i]^2}{n\sum_{i=1}^{n}((\frac{1}{s^i})W^i)^2} \tag{6}$$

W^i is the cumulative wealth received by node i; s^i is the "importance" score. The index formula is subject to the interpretation that it reports the portion of the population to which the attribution has been fair [1]. For instance, if \$10 should be equally distributed among a population of 10 people, but instead, only 2 people receive \$5 each, while the rest receives \$0, the Fairness Index is 0.2 (out of 1.0). This is how the Fairness Index provides a way to measure fairness.

Mathematically, F is simply the square of the components of a vector, $\mathbf{v}$, where $\mathbf{v} = (\frac{W^1}{s^1}, \frac{W^2}{s^2}, \ldots, \frac{W^n}{s^n})' / \sum_{i=1}^{n}\frac{W^i}{s^i}$ is an element in the simplex – see section III-B. This means that there exists a maximum value of $F = \frac{1}{n\sum_i^n {v^i}^2}$ with respect to $\mathbf{s}$, since $\sum_i^n (v^i)^2$ has a minimum.

It is possible that, although the cumulative wealth achieved is proportional or nearly proportional to the "importance" score, networks have a tendency to favour nodes that are already "important". This may result in a distribution of wealth that can be very skewed and unequal [4]. This is an occurrence of the power law, which suggests that the shape of wealth distribution resembles the function below.

$$p(x) = Cx^{-\alpha}, \quad x > x_{min} \tag{7}$$

$\alpha > 1$, typically $2 < \alpha < 3$, is a single known parameter and C is the normalising constant. This distribution may be identified as a Pareto distribution. The *Power Law* has much heavier tails than an exponential distribution, such that certain events, which are impossible in the exponential or truncated normal distribution, are possible for the *Power Law*. It is well known that the power law tends to manifest frequently for web-based networks, when the in-degree distribution of a graph is considered [10].

The inequality of wealth can be observed graphically through a Lorenz Curve [4], [11], which considers the relationship between the fraction of nodes and the fraction of degree value. The closer α is to 2[1], the more wealth is controlled by the smaller fraction. The implication of this is that "the rich get richer", hence the need for a global reward to counter-balance the effects of the power law. We establish that we want the final distribution of wealth, after all rewards have been allocated, to follow some predecided distribution. This is different from measuring how much an entitlement deviates from cumulative wealth when discussing the Fairness Index. A more recent proposal of this is a derivation based on entropy [2], [3]. Venkat proposed the use of an information-theoretic definition of fairness, concluding that the least biased distribution is the log-normal one, since it is the distribution with the highest entropy, given some knowledge of fixed parameters [2], [3]. The fixed parameters are chosen arbitrarily: $\sigma = \frac{R}{2a}$ and μ, where R is the range of wealth values and a is the significance level. The σ parameter defines the desired standard deviation for the distribution of our wealth and μ defines the average wealth [2]. The distribution has a positive support, there exists an average wealth that is closer to its median (as compared to a power law), and there are no probable extreme values outside of the specified range. We expect that, after all the rewards have been offered to the network, the wealth distribution in each reward period reflects the shape of the log-normal function with parameters μ and σ.

G. Problem

We use the terms defined in the previous sections to state the problem here. There exist n nodes in the network ξ, which is the result of the union of m different trees, $\{\mathcal{G}^i\}_i$. There exist two reward types, i.e. many local rewards and a global reward. $\mathbf{p}$ corresponds to a branch attribution concerning a local reward, whereas w^i is the weight that corresponds to the global reward. s^i is the "importance" score. In a particular reward cycle t, we provide the network with all the local rewards first. The

[1] $\alpha > 2$ to ensure that the integrals in the estimate of the fraction of wealth converge

[2] μ and σ are the standard normal parameters after the log operation is applied.

wealth received by a particular node may be cumulative, because the node receives wealth for both being the root node of a tree and being in the path of other trees. Once we have observed this distribution of local rewards, the global reward follows. We state the problems as below:

1) Consider a particular node x^i, such that $x^i \in V(\mathcal{G}^j).\forall j$. The cumulative wealth it obtains through local rewards is $W^i = \sum_j W^{j\cdots}$. How can we assign all the branch attributions in the network $\{\mathbf{p^{ijk}}\}_{ijk}$, such that all scores are proportional to its cumulative wealth? In other words, $s^i \propto \sum_j W^i = W^{j\cdots}$.
2) If the "importance" score follows a power-law distribution, how can we dampen the effects of the power law, by intervening at each branch in the network?
3) After all the local rewards have been delivered – given that we know $\{\mathbf{p^{ijk}}\}_{ijk}$ and the cumulative wealth distribution after all the local rewards – how can we choose $\{w^i\}_i$, such that the final cumulative wealth distribution after the local and global rewards is log-normal, with chosen parameters μ and σ?

The key problem: Although we have an impression of each node's "importance" in the global scope, a node obtains different rewards from different trees and we must transpose our global "importance" score into a list of branch attributions.

III. Mathematical Tools

A. TrustRank

In this section, we introduce the *PageRank* algorithm, a popularly cited algorithm in the literature of centrality measures. The purpose of centrality measures and *PageRank* is to identify the "important" or "central" nodes in the network, by giving each node a score [12], [13]. Our proposed model uses these *PageRank* scores to derive attributions for all local rewards. Although our algorithm follows the same underlying principles as PageRank, we use the term TrustRank to refer to this algorithm, because we propagate trust in a node and deal with nodes in a network instead of web pages. In section V-A, we use a specific application of TrustRank to remove spam nodes[3].

TrustRank is an iterative eigenvector centrality measure, meaning that the solution of node scores corresponds to the dominant eigenvector [4] of some encoding or adjacency matrix – more specifically, the left eigenvector [12]. Below is the transition or trust matrix:

$$T = \alpha(A + H) + \frac{(1-\alpha)}{n}\mathbf{11'} \tag{8}$$

A is the adjacency matrix; H is a matrix with columns of $\frac{1}{n}$, which correspond to the columns of A that consist of zeroes, while the other entries are zeroes; α is the random surfing coefficient. T has certain properties – it is stochastic, aperiodic, and irreducible (this is equivalent to the definition of primitive) – ensuring that the power method, an iterative method for finding the dominant eigenvector, $\mathbf{s}$, converges [12]. The original interpretation of the *TrustRank* solution is that, under the random surfer model, s^i represents the fraction of time spent on a page (the sum of *TrustRank* scores is 1) and that pages are able to upvote other pages by adding backlinks, which act as a good signal for "good" pages; it is even better, if those upvotes originate from other "good" pages [8]. This is a natural solution for any reputation-based system. In our context, the nodes follow the same principle, according to which a directed edge is seen as an upvote. It is a general convention that a node with more edges pointing to it is scored higher, whereas a node with more edges pointing out of it tends to spread out it's score more widely.

There are several reasons why *TrustRank* became the chosen centrality measure. Firstly, certain measures, such as in-degree centrality, only record first-degree influence; *TrustRank* accounts for global influence. Secondly, certain parameters are adjustable, to account for the scoring of a particular node. For example, $\mathbf{11'}$ in equation 8 can be replaced by fudge factor $1\mathbf{z'}$; the teleportation probabilities are no longer uniform. Thirdly, *TrustRank* offers an interpretation of the probability of activating a particular node. This enables us to interpret the "importance" of every node as a composition. Compositions are essential, because they make it possible for us to manipulate sub-compositions, a subset of the nodes, and preserve certain notions of geometry and interpretation – see section III-B.

B. Vector Space on Simplex

A vector of n real numbers is conventionally studied and evaluated in the Euclidean space, $\mathbb{R}$. Compositions exist in a different space, called the "simplex", which is a subset of the real space [22], [23].

$$S^n = \{\mathbf{p} \ : \ \sum_i^n p^i = 1, p^i > 0 \ \ \forall i\} \tag{9}$$

The simplex is a vector space under its own binary operations: $\oplus$ perturbation (summation) and $\otimes$ external operation (scalar multiplication). These operations allow us to construct notions of distance and the inner-product that have interpretations similar to the Euclidean Space, i.e. form a weighted average of two compositions, while accounting for natural collinearity [22]. The advantage of this is the preservation of sub-compositional coherence, an inherent property of the simplex, which ensures the interpretability of the sub-compositions [22]. To illustrate this, we define *closure* as the division of a vector by its sum; a typical transformation onto S^n from $\mathbb{R}$. We also define a sub-composition as a sub-vector of a composition

[3]The original paper that coined the term TrustRank focuses on spam filtering only, but we use the term more generally [18]

[4]the eigenvector corresponding to the largest eigenvalue

that is closed. An example of sub-compositional incoherence can be illustrated by using usual sum operations. We sum two compositions with the same dimensions, $\mathbf{p^A}$ and $\mathbf{p^B}$. The incoherence occurs, because a sub-composition of the sum is different from the result we would receive, if we summed respective sub-compositions of $\mathbf{p^A}$ and $\mathbf{p^B}$. This leads to severe consequences, if we use Euclidean distance, according to which a sub-composition may have a longer distance than the original composition.

Although this may not apply completely to all parts of our solution, we believe that, if we want to use the global *TrustRank*, it may be useful – especially, if we consider branch attributions as a sub-composition of the vector of *TrustRank* scores, a composition. For example, if we intend to form a weighted average of two compositions, we need to perturb, rather than sum, the two components – see section V-B. Otherwise, branch attributions will not be consistent; if a node is removed from a branch, for instance, the other nodes of that branch will be unsatisfied, because the removal of one node affects their proportional share of the wealth.

IV. Model

There are many components that govern this model, each with a specific aptitude for either satisfying evaluation metrics or conforming to some interpretation. A key assumption in our model is that the "importance" score represents the benchmark for estimating branch attributions. Below are proposed solutions to the problems introduced in section II-G, in the corresponding order.

1) This is the most difficult problem and may be unsolvable mathematically. In the local scope, we need to assign branch attributions. One way to do this is to pool all local, incoming wealth and simply base the rewards on the TrustRank score. This variant satisfies the requirement of proportional attribution, but has the disadvantage of not using a "wealth trickle" mechanism. Alternatively, we may want the branch attribution to be a sub-composition of the *TrustRank* scores. Assuming that all nodes have incoming wealth, important nodes – which have more paths to receive wealth from different nodes – would obtain more, cumulatively. This would worsen the skew of wealth, because those nodes would not only receive higher proportions in the branch attribution, but more wealth would flow through them as well. Thirdly, we can assume that each branch attribution is divided equally, which appears sensible too, since "important" nodes would naturally receive more wealth. Furthermore, the idea that each branch attribution is divided equally is an easily interpretable concept. This is the equivalent of taking a sub-composition of a branch, when *TrustRank* is run on a tree. A fourth option would be to turn the problem into a constraint optimisation problem, whereby the branch attributions would be estimated with the constraints that they sum to 1 and the fairness function as the objective function is maximised as below.

$$\max_{\{\mathbf{p^{ijk}}\}_{ijk}} F_{index} \tag{10}$$

$$\sum_l p^{ijkl} = 1 \quad \forall i, j, k \tag{11}$$

Each W in the fairness index is a function of $\mathbf{p}$'s. This stands in contrast to the first three solutions, which may require data to verify that they work, since the amount of wealth a root node obtains – if it obtains any wealth at all – is random. Therefore, it is impossible to predict whether those variables, along with our suggested rules of branch attributions, fulfil the criteria of proportionality. In comparison, this solution only requires the optimisation to be accurate. But obviously, difficulty is incurred in the optimisation problem as well – in terms of whether the global maximum is obtained or not or whether it will be settled with a solution that excludes particular nodes from receiving any wealth at all.

2) The TrustRank scores and, correspondingly, the cumulative wealth of nodes in a web-type network will follow a power law distribution. Although we cannot confirm this empirically, we know approximately that – assuming that the optimisation solution is achieved – this will be the case, since the tails of *TrustRank* have some equivalence with in-degree distributions, which, as we know, will follow a power law distribution [13]. A case may arise where maintaining proportionality between the TrustRank scores and the cumulative wealth of nodes is of less interest; for example, if we do not want to avoid the adverse effects of wealth inequality, which are the consequence of the power law distribution. In that case, we could weigh each branch attribution artificially, by using information about the number of in-degree scores or *TrustRank* scores. Similar to the global reward, intervention would occur in the local scope to reduce the inequality of wealth, except that the wealthy nodes, in this case, would be penalized. We would utilise a weighting function that is either sigmoid or exponentially shaped, such that nodes with smaller scores are weighted higher and nodes with higher scores are weighted lower. After multiplying a branch attribution with its weights, we would transform it back into a composition. An example of an exponentially shaped function for weighting is shown below.

$$f(s^i) = e^{-\lambda s^i} \tag{12}$$

λ is the shape parameter that can be adjusted or determined by some property of the branch attribution or the "importance" of the parent node.

3) Assuming that we obtain a power law form for the distribution of wealth, we can transform it into a log-normal distribution, using the transformation below.

$$W' = f(W) = \exp\left(\mu + \sigma\,\Phi^{-1}\left(1 - \left(\frac{W_{\min}}{W}\right)^{\alpha}\right)\right) \tag{13}$$

Φ is the standard cumulative normal distribution; W is a data point from the power law distribution; W' is the transformed data point that is log-normal. μ and σ are the parameters of the log-normal distribution (on the log scale); x_{min} and α are the parameters of the power law (or Pareto distribution). Through this transformation, we can derive the wealth that needs to be added to or subtracted from every node, to obtain a log-normal distribution. Additionally, we impose the rules below on the global reward.

   a) The ranking of the wealth of nodes does not change. The global reward does not improve a node's ranking.
   b) Wealth can only be added to and never subtracted from a node.

V. Adversarial Measures

We must note a particular concern when developing this model. Since the model involves allocating a reward to nodes, there are potential issues regarding adversarial gaming within the network.

This is a recurrent topic in the field of search engines, where pages compete to reserve a higher ranking and can be referred to as spam. For example, we see this in the occurrence of "link farms". A link farm is a set of nodes, linked together, that intends to boost its "importance" within the hierarchy of nodes. We investigate this on the premise that the attacker seeks a high *TrustRank* score, although this is not necessarily the only motivation. In our model, we assume that particular nodes can receive wealth without incurring the cost of creation for a node and that the branch attributions are estimated using our optimisation.

We highlight the most obvious ways in which our reward model may be exploited.

1) Artificial linking to a node close to a root node, such that it obtains the highest score.
2) Parents creating nodes as their children, to siphon the wealth that is passed down from them. This occurs because there is no cost of creation for a node.

In our model, nodes closer to wealth nodes tend to obtain a larger portion of the rewards, since our *wealth trickle* works in a hierarchical structure. Therefore, most of the monitoring should occur near those wealth nodes. An advantage of *TrustRank* is that the votes given to a node by another node depend on not only the node's act of giving the vote, but also on the quality of the node itself, i.e. on the number of votes the node itself receives. Therefore, a very large number of artificial nodes has to be created, in order to influence a node's *TrustRank* score – which makes unusual behaviour observable [14].

A. Spam Filtering

TrustRank can be turned into a method to combat spam nodes. The method is based on the assumption that it is unlikely for a non-spam node to point towards a spam node. A small seed set of nodes in the network is identified as trustworthy and judged by an external scorer, based on observation of the nodes' properties, e.g. their GitHub profile. Then, Personalised PageRank/TrustRank is run, using the trusted nodes as its teleportation set, with uniform weights within the set [18]. This teleportation set can be modified to have non-uniform weights, in order to bias nodes with higher trust scores too. The amount of trust is reduced based on a node's distance from a trustworthy node; therefore, nodes that are not upvoted by trustworthy nodes will have a lower score. Furthermore, trust is dispersed: A node that is pointed to by a non-spam node with few outward-pointing links will be more trusted than a node pointed to by a non-spam node with many outward-pointing links. Any node with a score lower than a particular threshold will be removed from the reward distribution. Additionally, there is an innovation of TrustRank called Anti-TrustRank, which has been researched as well: It simply chooses seeds of spam nodes and propagates anti-trust as opposed to trust. This other method is based on the assumption that nodes pointing to spam nodes are likely to be spam nodes as well [15]. Therefore, nodes with high anti-trust scores will be categorised as spam.

Although we do not examine it in detail, this method can be considered, because we would merely need to check for spam in nodes that are close to root nodes, as they are the nodes that accumulate the biggest amount of wealth; root nodes themselves are unlikely to be spam nodes. Therefore, our seed set should stem from nodes close to root nodes. An obvious concern is that, if spam penalises a node that is artificially upvoted, it is possible for nodes to carry out attacks on other nodes. This can easily and very possibly occur in our model – similar to a denial-of-service-attack, which uses unwanted traffic to the detriment of the legitimate user. Literature makes it clear that spam detection is no easy task; and although many methods have been suggested, all are relatively approximate. The methods suggested by literature tend to favour a large amount of nodes, to cater to the nature of the web – the majority of which is spam content. At least, we do not expect our network to be as spam-infested.

Once exploited nodes have been removed, we can robustify the wealth signals through a technique called "Branch Filtering". Parents can voice their opinions on the size of a branch attribution, which are compared to the estimated branch attribution, to yield a weighted estimate – see section V-B.

B. Branch Filtering

Estimates of branch attributions from the network may be noisy. Therefore, recommendations from the parents can provide a better signal regarding the children's entitlements. Filtering is a method to weigh out branch attribution scores (by estimation) and the branch attribution recommended by the parent node, which is known as the user attribution.

We introduce the notation of the user attribution $\mathbf{u}^{ijk}$ and the filtered branch attribution $\hat{\mathbf{p}}^{ijk}$. η represents the filtering function. The branch attribution $\mathbf{p}^{ijk}$ appears as before, being determined by the estimation of $\mathbf{p}$, with respect to the objective function. We add indexes that represent the states at different times, denoted by t. All these vectors are compositions.

$$\eta(\mathbf{u_t^{ijk}}, \mathbf{p_t^{ijk}}) = \hat{\mathbf{p}}_\mathbf{t}^\mathbf{ijk} \tag{14}$$

The filtered branch attribution can be described as the best estimate of the correct branch attribution, according to historical data. The rules that govern the filtering formula derived and described below are as follows:

1) **Missing User's Opinion**: If the user attribution is non-existent at time t, the filtered attribution at time $t-1$ (or the branch attribution at time t, if the former is unavailable) is the best estimate.
2) **Agreement**: If the user attribution and the branch attribution agree at time t, the filtered branch attribution should be confirmed with low variance and have strong inertia. When the user attribution at time t agrees with the previous filtered attribution at time $t-1$, it is automatically the best estimate of the branch attribution.
3) **Disagreement**: If the user attribution and the branch attribution disagree at time t, the best estimate of the branch attribution is a weighted average between the filtered branch attribution at time $t-1$ and the user attribution.
4) **Volatility**: Any past and frequent disagreements between branch attributions and user attributions, as well as between branch attributions (or user attributions) and previous filtered branch attributions, will make the best estimate uncertain. Based on past data, the weights encode the preference, determining whether to prefer the filtered branch attributions at time $t-1$ or the user attributions at time t.

These rules ensure that neither the user nor the network structure has full autonomy in determining the estimated composition. We can easily form a piecewise formula of $\hat{\mathbf{p}}_\mathbf{t}^\mathbf{ijk}$, as weighting elements $\mathbf{u_t^{ijk}}$,and $\hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk}$.

$$\hat{\mathbf{p}}_\mathbf{t}^\mathbf{ijk} = \begin{cases} \hat{\mathbf{p}}_{t-1}^{ijk}(OR\ \mathbf{p}_t^{ijk}), & \text{if condition 1} \\ \mathbf{u}_t^{ijk},\ \beta_t = \beta_{t-1} - 0.05 & \text{if condition 2a} \\ \mathbf{p}_t^{ijk},\ \beta_t = \beta_{t-1} + 0.05 & \text{if condition 2b} \\ \mathbf{u}_t^{ijk} & \text{if condition 2c} \\ \bar{\mathbf{p}}_t^{ijk},\ \beta_t = \beta_{t-1} - 0.025 & \text{if condition 3a} \\ \bar{\mathbf{p}}_t^{ijk},\ \beta_t = \beta_{t-1} + 0.025 & \text{if condition 3b} \\ \bar{\mathbf{p}}_t^{ijk} & \text{Otherwise} \end{cases}$$

where $\bar{p}_t^{ijk} = \beta_t \odot \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk} \oplus (1-\beta_t) \odot \mathbf{u_t^{ijk}}$. The conditions are listed below.

1) Condition 1: $\mathbf{u_t^{ijk}}$ is missing
2) Condition 2: $T(d(\mathbf{p_t^{ijk}} \ominus \mathbf{u_t^{ijk}})) < \kappa_3$
   a) AND $T(d(\mathbf{u_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_2$ AND $T(d(\mathbf{p_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) > \kappa_1$
   b) AND $T(d(\mathbf{u_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) > \kappa_2$ AND $T(d(\mathbf{p_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_1$
   c) AND $T(d(\mathbf{u_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_2$ AND $T(d(\mathbf{p_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_1$
3) Condition 3: $T(d(\mathbf{p_t^{ijk}} \ominus \mathbf{u_t^{ijk}})) > \kappa_3$
   a) AND $T(d(\mathbf{u_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_2$ AND $T(d(\mathbf{p_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) > \kappa_1$
   b) AND $T(d(\mathbf{u_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) > \kappa_2$ AND $T(d(\mathbf{p_t^{ijk}} \ominus \hat{\mathbf{p}}_\mathbf{t-1}^\mathbf{ijk})) < \kappa_1$

d is the distance operator defined on the simplex; β_t is a weighted number; T is a sigmoid-shaped mapping from the $\mathbb{R}^+$ to (0,1). κ_1, κ_2, and κ_3 are constants in $(0,1)$. There is good reason to use the simplex vector space. In a practical setting, there may arise a problem, whereby a particular node is deemed unworthy of a reward and has to be removed. To account for the change in dimension from time $t-1$ to t, we need only consider a sub-composition of the $\hat{p}_{t-1}^{ijk}$ to match the dimensions of $\hat{p}_t^{ijk}$ and $\hat{u}_t^{ijk}$. There will be no inconsistency in the proportions that would otherwise be held by a particular node because sub-compositional coherence is preserved [22]. In the case where the dimensions of the attributions increase, we resolve this by turning $\mathbf{p_{t-1}^{ijk}}$ into $\mathbf{p_t^{ijk}}$. This situation is not ideal because the same coherence for a reduce in dimension is not preserved. However, it can be argued that this change is induced by the parent and not the spam filter.

As we construct our filtering function, placing trust in the hands of the parent nodes can also invite opportunities for gaming the system. A worrying example is a parent's option to recommend that all the wealth in the branch be given to a particular child node. Therefore, we also weigh in the past history of agreement between the user attribution and the branch attribution, to reach the most informed composition, as shown by the third condition.

Further work in filtering can lie in applying the Kalman filter [24]. A Kalman filter is a state-of-the-art filter on time series that has some normality assumption

[5]. However, a Kalman filter operates in the real dimension; but this can be resolved by using isometric log-ratio coordinates, which preserve all the elegant properties of the simplex vector space we have defined. When using the Kalman filter, an online parameter estimation of variances will aid in the incorporation of inertia and weighting in the time series, i.e. if the user attribution and the branch attribution agree, the variance of the branch attribution will be reduced and the filtered attribution will have a higher resilience in following the branch attribution.

VI. Further Work

This document's objective is to specify the problem and orientate the framework; although many unanswered details remain. Much additional work must be commissioned for further investigation of the problem.

A. Research Notes

This section explicitly records the uncertainty of the author and the risks with regards to the philosophy and implementability of the methods. We highlight possible research directions.

- **Optimisation** Optimisation with *many* equality constraints can be a considerable problem. The fairness metric is not a convex function; therefore, it is uncertain if a global optimum can be achieved [27]. In fact, there is a good chance that estimation of branch attributions is not best performed through optimisation. We stated in section IV that – as an alternative to optimisation – we can employ simple rules in determining the branch attribution. However, the question of whether, through these simple rules, wealth is distributed proportionally, based on "importance", cannot be answered without empirical verification. The investigation may be continued as data becomes available – see seventh bullet point.
- **Global Importance** TrustRank discovers the "importance" of a node globally, but not locally. This introduces the debate on whether or not nodes should be allocated local rewards based on their global "importance". Our adjustments, such as weighting of the branch attribution and filtering, make it difficult to anticipate if the proportionality with TrustRank scores will be achieved – which begs the question: Are the TrustRank scores even relevant, when the signal of "importance" derived from them becomes overshadowed, as nodes interact with one another?
- **Adversarial Measures** The filtering and TrustRank solutions are relatively adversarial [18], [21]. Since we do not impose stringent rules on the participation in the network and the creation of links within the nodes, this may give rise to the emergence of spam nodes. Therefore, in a typical application, the adversarial issues have to be given attention, including careful consideration of the way in which nodes participate in the network. Additionally, the solutions by Anti-TrustRank would not only require TrustRank scores to be re-computed (which is expensive), but also involve monitoring by particular authorities who are accustomed to identifying spam, leading to overhead costs [15]. Apart from the simple rules with available data, we have not explored any game-theoretic methods that may be useful, such as approaches to resource allocation networks, which would enable us to understand the incentives and equilibria involved in inter-node relationships [29], [30]. Within the sub-field, the evolutionary game-theoretic perspective may be utilised, when it comes to nodes that are acting in not only a profit-maximising fashion, but a charitable fashion [40].
- **Global Reward** Although the global reward identifies the discrepancy between the wealth of nodes under the power law and the log-normal distribution, it does not promise that we transform the data sufficiently to reach a log-normal distribution, especially if the global rewards are limited. In fact, we are skeptical that it is possible to accurately map two distributions of different shape. Therefore, an approximate solution that is not mathematically verifiable has to be devised to distribute wealth. In this case, partitioning wealth nodes into classes and rewarding corresponding classes a portion of the global reward pool may be a possible solution although not all the rules mentioned in section IV may be fulfilled, i.e. the ranking of nodes is not preserved.
- **Complexity** Computational complexity is also a concern. Although it is possible for all methods to run at computationally feasible times, the simultaneous computation of TrustRank, Anti-TrustRank, wealth trickle, optimisation, and filtering – in order to produce all the branch attributions – may take too long, depending on the application at hand. A possible solution is to partition the computational load, such that this work can be performed in a distributed manner. For example, many efforts have been made to speed up the computation of a Monte Carlo approximation of TrustRank by distributing work using Hadoop MapReduce [28]. The filtering process is easily parallelisable, because the operations resemble linear algebraic operations, which is typical in parallelisation through particle filters or Kalman filters [34], [35]. Regarding the partitioning of the workload, there is well-studied literature that suggests ways of getting different servers to communicate with each other through consensus

[5]A normality assumption is not typically important. Otherwise, options like the unscented or extended Kalman filter are available. The particle filter is another option, but the use of random number generation will make the application a computational challenge

algorithms, such that, even when particular servers fail, the majority of them reach a consensual decision [34].

- **Sequential Nature of Networks** Assuming that the application considers growing networks through time, in order to save computation, we want to avoid any re-computation and preserve existing interpretations. The compositional analysis of filtering, in section V-B, for example, correctly addresses a reduction in dimension of a particular branch attribution; but for an increase in dimension, the information filter has to be restarted, meaning that all past information is lost. It is well-known that TrustRank is difficult to compute sequentially too; and although several methods – such as aggregation – exist, they only provide approximate solutions [33].
- **Availability of Data** Perhaps the biggest game changer in our research will lie in the availability of user data – whether it is simulated or real data. It is possible to perform data simulation, resulting in a global TrustRank score, with or without power laws, via random graph generation models. Amongst these models are preferential attachment models [31] and the Watts-Strogatz Model [32]. We can then perform a breadth-first search to form a tree based on a randomly selected node, in order to accumulate our list of local tree structures, although not all of the trees need to be discovered. We can then fabricate wealth on each tree's root-node by simulating a distribution that is conditional on In-Degree or TrustRank scores. For real data, we plan to make use of the tree dependency structure in the "Node Package Manager" (NPM) registry as a way to evaluate our model. The NPM API readily provides a method to extract tree dependency, with each software project containing an identification index [37]. The only work required is in taking the union of trees, which is a well-known operation in any graph software. For example, *igraph* in R provides the function of union, as long as we maintain a unique identifier from each software project as a string [38]. The NPM dependency network is a large structure; therefore, it may be suitable for us to exclude particular software projects that are redundant via a sort of URL-type processing, which is common in the web-indexing literature. The availability of data enables us to confirm our model empirically. All the simple rules on branch attribution and filtering can be used and cross-checked with the TrustRank scores and the fairness metric.

B. *Proof-of-Analytics*

Decentralisation has been a recurring topic in terms of advancements in blockchain technology. We introduce a plausible application; our framework is applied to facilitate and complement applications on the blockchain, particularly on the Ethereum platform. The blockchain is a decentralised ledger that is maintained and secured by a distributed network of computers. In the Proof-of-Work protocol [6], the ledger is secured through a cryptographical hash, which ensures that data encoded into a block cannot be changed. Miners – who are participants in the network – compete with other miners to secure a particular block of transaction, by solving this difficult-to-compute cryptographical hash and obtain a reward from it, which is an incentive to keep the network secure. Any tampering with created blocks will be rejected by the network of nodes – both socially and economically – because only valid blocks are accepted, assuming that the majority of the hashing power originates from honest actors. Proof-of-Work has thus far been the gold standard in blockchain protocol algorithms, although this is not the only protocol that has been developed; others are Proof-of-Stake, Proof-of-Importance, Proof-of-Useful-Work, Proof-of-Burn, and many more. Extensive research has been trying to find the best protocol to reach a consensus within the network of nodes, such that adversarial intervention can be depleted and centralisation is impossible.

One of the popular platforms that execute their own blockchain is known as Ethereum. Ethereum is a codebase that runs a blockchain under the Proof-of-Work protocol, although there are plans to switch to Proof-of-Stake in the near future. Ethereum has successfully applied smart contracts to form contractual agreements between peers in the blockchain, but there are several issues concerning the practical implementation of smart contracts: The expensive gas prices make it infeasible to perform recursive computation in any smart-contract code, data is not readily available within the blockchain network, verification is done by ALL nodes in the network (which requires a lot of computation), and the API for external data has to be verified to ensure that the data comes from a trusted source. A popularly proposed solution for these problems is to enable the use of oracles. Oracles are simply middleware that can interact with the blockchain; they can be used for computationally intensive work or to fetch data from external data sources, hence enriching the application space that would otherwise be restrictive [39]. Until the development of further blockchain protocols to deal with heavy computational systems, applications are only tractable when we use a kind of oracle. We propose a type of Proof-of-Concept, existing off-chain, that enables us to utilise both the underlying smart contract framework on Ethereum and oracles in our distributed network, such that we can compute proportional attributions and allocations in an honest and verifiable fashion. We attempt to add the feature of distributed consensus to our model, which could otherwise be exploited through the central server's creation of a single point of failure. This protocol is

[6]This protocol originates from the ideology of Satoshi Nakamoto and is named Proof-of-Work.

called Proof-of-Analytics; the idea is to perform network analytics, using our specified implementation on the graph network and users' opinions. We can then validate the proportional attribution, based on an honest majority vote of the trusted oracles, using a statistically significant assessment. Note: This sections serves as an example of an application of the proportional attribution model(PAAM), but is not a rigorous treatment to Proof-of-Analytics. Proof-of-Analytics will be formally introduces with Proof-of-Usage and Proof-of-Exercise in The Cardstack Consensus paper (to be publised).

We describe a plausible block cycle as follows: A set of nodes[7] is selected at random, based on the previous block hash in the reward smart contract interfacing with these distributed nodes; this is called *cryptographic sortition*, familiarly coined in *Algorand* [25]. All selected nodes perform the full computation described in our paper (with similar software) and can be referred to as *validators* of each block cycle. A great advantage is the fact that none of the *validators* is excluded from the computational cost or setup, because the barriers to the network are low, i.e. users can participate by simply downloading the software on their computers. A smart contract then reads the results reported by each *validator* and assesses them, using a statistically significant assessment, i.e. a measure of the discrepancy between the results. If the variance is low, the *validators* receive minting rewards. If the variance is high (above some specified threshold), certain nodes with different results – outliers – will be removed without being rewarded, to find out whether this leads to a drop in variance. If the variance of the results is still high afterwards, the smart contract compels more nodes to participate in the validation process and re-engages the computation from the start. Unlike Proof-of-Work, there is no race for obtaining the next block, which eliminates the influence of hardware-advantaged *validators* that would centralise the network – importance is placed on giving the correct answer. Yet, there will be a time limit; the *validators* have to report their results by a certain deadline to qualify for obtaining the reward.

This function can be compared to a teacher in a class room, who only gives homework to a randomly selected set of students, since he is too lazy to mark all the students' books. Each student is given the same reference book to solve the problems at hand. In fact, the teacher is not only lazy, but does not know the answers himself. Since he cannot answer the question he has asked, he compares the students' answers, analysing the difference between them. If a majority of students gives a particular answer, that answer is recorded as the correct one, assuming that the minority gives the wrong answer. If no majority is found, the teacher asks more students to do the homework. This protocol, similar to many others, relies on democratic correctness, i.e. the democratic choice of the juror must be correct, but removed from the usual subjectivity of a court of law. Additionally, the problem may be partitioned, such that TrustRank, Anti-TrustRank, optimisation, and filtering are different functionalities that cannot be carried out by the same node, so long as the selected nodes communicate reliably with one another.

On top of the computation performed by the *validators*, we engage a stake system, which requires another randomly selected set of nodes to vote for the *validators* that look like spam – these nodes are called *choosers*. This way, the responsibility of our spam oracle (as described in section V) – which uses Anti-TrustRank – can be divided among several actors of the network, instead of being controlled by a single authority who is governed by the developer. The correct votes will result in the *choosers* being rewarded, but without their stakes being lost if the results are wrong (so as not to take away their incentive to act as *choosers*). The *choosers* are given the facility to assess a node's profile subjectively or run Anti-TrustRank on its client, to make use of historical votes (each node will be assigned a trust score that is updated through these votes); this allows them to make an informed decision about which nodes are spam – therefore, it is at the *choosers'* discretion. Obviously, if there are no spam nodes as *validators*, a *chooser* can refrain from voting. In this system, the network can act as a community to pinpoint spam nodes, thereby removing the possibility of those spam nodes being randomly selected as *validators* and thus keeping them from collecting rewards. Any node can participate in this system, since we assume that a node's entry checkpoint requires it to have a stake in the system already. Besides: Would a dishonest actor be able to race to complete these computations and collect the rewards? Assuming that the network begins with a majority of honest actors, random selection reduces the probability of choosing a dishonest actor. The root nodes of the tree are also participants in the network; they have invested a large sum of money and, therefore, have no incentive to vote for spam nodes that have not contributed to the network's success.

Above, we have described a plausible application of the reward model, although details of the implementation have been left aside. This section exemplifies that proportional attribution and allocation should be used in some area of industry and practical application. We believe that we are proposing a method of democratising the web, which is similar to the way in which Google or Facebook use users' opinions in a partial P2P fashion to upvote others based on reputation, but with active participation of the network. Our implementation typically resides off-chain, but the proportional attribution uses the same paradigms as stated. The potential of the proportional attribution gives hope to a future implementation of reputation on the blockchain, where the central use of oracles is removed and the platforms stray

[7]We use the word "node" to replace the word "oracle", because there exists no distinction in our example.

away from computationally intensive protocols.

VII. CONCLUSIONS

The framework we have proposed is very extensive and requires an accumulation of different mathematical tools to solve the problem. We have seen that ”importance” can be attributed proportionally through local changes and an estimation of branch attributions. The distribution of rewards can be transformed from being unequal to following a log-normal distribution, which restricts the nature of wealth inequality. Filtering and other adversarial measures can be used as means to prevent players from gaming the system and selfishly obtaining more rewards than they deserve. This framework is the first step in the direction of rewarding open-source projects in networks. We believe that, if we can develop a stronger foundation regarding adversarial concerns, networks can build trust in the distribution of wealth that is realised in this particular manner. The expansion of the utility of assessing how much a node deserves is not only based on some global ”importance” scores, but also on recommendations from the parent nodes, and can be fruitful in applied areas, such as blockchain technology. Previous research has shown that, although the blockchain is well suited for secure ledger accounting, it is reasonably inefficient in performing any other action. Therefore, actions with hard complexity, like the attribution of wealth, are unreliable. Most blockchain applications have been carrying out the functions of applications outside of the blockchain through oracle functions, which renders the applications more insecure. However, our framework may be associated with the use of the idea called Proof-of-Analytics, which allows a collection of oracle functions to be operated by any party, in order to confirm how much each node deserves through proportional attribution. Finally, since networks are already correlated and demonstrate some algebraic properties, we believe that our use of the simplex vector space may be a step towards generalising the division of wealth too, which has been approached in a flat geometry.

APPENDIX

Suppose $\mathbf{x}$ and $\mathbf{y}$ are compositions and α is a scalar.

$$\mathbf{x} \oplus \mathbf{y} = C[x_1y_1, x_2y_2, \ldots, x_py_p] \tag{15}$$

$$\alpha \odot \mathbf{y} = C[x_1^\alpha, x_2^\alpha, \ldots, x_p^\alpha] \tag{16}$$

ACKNOWLEDGMENT

Thank you to all the authors of previous material whose intellectual contributions have made this paper possible

References

[1] Jain, Raj, Dah-Ming Chiu, and William R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Vol. 38. Hudson, MA: Eastern Research Laboratory, Digital Equipment Corporation, 1984. APA

[2] Venkatasubramanian, Venkat. ”What is fair pay for executives? An information theoretic analysis of wage distributions.” Entropy 11.4 (2009): 766-781. APA

[3] Venkatasubramanian, Venkat, Yu Luo, and Jay Sethuraman. ”How much inequality in income is fair? A microeconomic game theoretic perspective.” Physica A: Statistical Mechanics and its Applications 435 (2015): 120-138.

[4] Newman, Mark EJ. ”Power laws, Pareto distributions and Zipf's law.” Contemporary physics 46.5 (2005): 323-351. APA

[5] Ramos, Juan. ”Using tf-idf to determine word relevance in document queries.” Proceedings of the first instructional conference on machine learning. Vol. 242. 2003. APA

[6] Brams, Steven J., and Alan D. Taylor. Fair Division: From cake-cutting to dispute resolution. Cambridge University Press, 1996.

[7] Landherr, Andrea, Bettina Friedl, and Julia Heidemann. ”A critical review of centrality measures in social networks.” Business & Information Systems Engineering 2.6 (2010): 371-385.

[8] Page, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab, 1999. APA

[9] Kleinberg, Jon M. ”Authoritative sources in a hyperlinked environment.” Journal of the ACM (JACM) 46.5 (1999): 604-632.

[10] Faloutsos, Michalis, Petros Faloutsos, and Christos Faloutsos. ”On power-law relationships of the internet topology.” ACM SIGCOMM computer communication review. Vol. 29. No. 4. ACM, 1999.

[11] Lorenz, Max O. ”Methods of measuring the concentration of wealth.” Publications of the American statistical association 9.70 (1905): 209-219.

[12] Page, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab, 1999.

[13] Litvak, Nelly, Werner RW Scheinhardt, and Yana Volkovich. ”In-degree and PageRank: why do they follow similar power laws?.” Internet mathematics 4.2-3 (2007): 175-198.

[14] Fetterly, Dennis, Mark Manasse, and Marc Najork. ”Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages.” Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004. ACM, 2004.

[15] Krishnan, Vijay, and Rashmi Raj. ”Web spam detection with anti-trust rank.” AIRWeb. Vol. 6. 2006.

[16] Benczur, Andras A., et al. ”SpamrankâĂŞfully automatic link spam detection work in progress.” Proceedings of the first international workshop on adversarial information retrieval on the web. 2005.

[17] Haveliwala, Taher H. ”Topic-sensitive pagerank.” Proceedings of the 11th international conference on World Wide Web. ACM, 2002.

[18] Gyongyi, Zoltan, Hector Garcia-Molina, and Jan Pedersen. ”Combating web spam with trustrank.” Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004.

[19] Jeh, Glen, and Jennifer Widom. ”Scaling personalized web search.” Proceedings of the 12th international conference on World Wide Web. ACM, 2003.

[20] Fogaras, Daniel, and Balazs Racz. ”Towards scaling fully personalized pagerank.” Algorithms and Models for the Web-Graph (2004): 105-117.

[21] Castillo, Carlos, and Brian D. Davison. ”Adversarial web search.” Foundations and TrendsÂő in Information Retrieval 4.5 (2011): 377-486.

[22] Pawlowsky-Glahn, Vera, Juan Jose Egozcue, and Raimon Tolosana Delgado. ”Lecture notes on compositional data analysis.” (2007).

[23] Aitchison, John. ”The statistical analysis of compositional data.” (1986).

[24] Mills, Terence C. ”Forecasting compositional time series.” Quality & Quantity 44.4 (2010): 673-690.

[25] Micali, Silvio. "ALGORAND: the efficient and democratic ledger." arXiv preprint arXiv:1607.01341 (2016).

[26] Kamvar, Sepandar D., Mario T. Schlosser, and Hector Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks." Proceedings of the 12th international conference on World Wide Web. ACM, 2003.

[27] Soriano, J. M. "Global minimum point of a convex function." Applied mathematics and computation 55.2 (1993): 213-218.

[28] Bahmani, Bahman, Kaushik Chakrabarti, and Dong Xin. "Fast personalized pagerank on mapreduce." Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011.

[29] Han, Zhu. Game theory in wireless and communication networks: theory, models, and applications. Cambridge University Press, 2012.

[30] Felegyhazi, Mark, and Jean-Pierre Hubaux. Game theory in wireless networks: A tutorial. No. LCA-REPORT-2006-002. 2006.

[31] Barabasi, Albert-Laszlo, and Reka Albert. "Emergence of scaling in random networks." science 286.5439 (1999): 509-512.

[32] Watts, Duncan J., and Steven H. Strogatz. "Collective dynamics of'small-world'networks." nature 393.6684 (1998): 440.

[33] Langville, Amy N., and Carl D. Meyer. Google's PageRank and beyond: The science of search engine rankings. Princeton University Press, 2011.

[34] Ongaro, Diego, and John K. Ousterhout. "In search of an understandable consensus algorithm." USENIX Annual Technical Conference. 2014.

[35] Brun, Olivier, Vincent Teuliere, and Jean-Marie Garcia. "Parallel particle filtering." Journal of Parallel and Distributed Computing 62.7 (2002): 1186-1202.

[36] Hashemipour, Hamid R., Sumit Roy, and Alan J. Laub. "Decentralized structures for parallel Kalman filtering." IEEE Transactions on Automatic Control 33.1 (1988): 88-94

[37] "These are the docs you're looking for." Npm Documentation, docs.npmjs.com/. Accessed 1 Aug. 2017. URL link to NPM documentation

[38] "R igraph manual pages." Igraph R manual pages, igraph.org/r/doc/. Accessed 11 Aug. 2017. URL link to igraph R documentation

[39] "Ethereum and Oracles." Ethereum Blog, 22 July 2014, blog.ethereum.org/2014/07/22/ethereum-and-oracles/. Accessed 10 Aug. 2017.

[40] Weibull, Jorgen W. Evolutionary game theory. MIT press, 1997.