Designing a Subjective Transitive Reputation Algorithm

Jacob Kirmayer

jacob@eas.foundation

Ethereum Attestation Service

July 22, 2024

Abstract

In decentralized networks, managing trust remains a pivotal challenge, particularly in environments where interactions are not governed by a central authority. This paper introduces an algorithm designed to compute and propagate trust in a decentralized manner by leveraging the concept of relative trust between nodes. Unlike traditional trust models that rely on absolute scores, our algorithm calculates trust based on the relative reliability of nodes as perceived by a given node in the network. Our introduction of this algorithm highlights the potential of decentralized relative trust mechanisms to enhance security and reliability in peer-to-peer and other distributed systems.

1 Scope and Design Criteria

Before introducing an algorithm to compute trust, we must define constraints for our algorithm to satisfy. Firstly, this algorithm will compute a relative trust score for any peer from the perspective of any other peer. Our algorithm will use peer-to-peer attestations of trust as a primitive to derive these scores. Since this process has no centralized arbiter, it must be the case that peers in the network have no incentive to attest positively to peers that they do not see as trustworthy. It must similarly be the case that peers have no disincentive that might discourage them from attesting to a trustworthy friend in the network. Crucially, our algorithm must not allow any peer to increase it's own reputation. Additionally, our algorithm must compute scores in a way that is reflective of trustworthiness. The criteria below summarize these considerations.

1. Non-Destructiveness. Every attestation of trust from one peer to another should result in the recipient's trust score increasing or remaining the same from the perspective of every other peer in the network.
2. Sybil-resistance and robustness towards malicious collectives. No peer should be able to create another peer whose reputation is greater than that of the peer itself, and no collective should be able to create a peer whose reputation is greater than that of the collective's most reputable member.
3. Balanced incentive structure. No peer should be able to raise or lower its own trust score from the perspective of any other peer. Furthermore, no peer should be able to alter its own standing relative to another peer from the perspective of a third observer.

2 Related Work

We separate these algorithms into two categories. Those that treat reputation as a subjective value and those that treat it as a global value. EigenTrust is a well-known algorithm of the first type, while Appleseed and Tidal trust are well-known algorithms of the second type.

2.1 EigenTrust

The first well-known algorithm belonging to this family is EigenTrust [3]. On its own, EigenTrust does not satisfy design constraint 1. Analysis of the EigenTrust algorithm yields many counterexamples where peers in a network are incentivized to exploit cycles in the trust graph to gain reputation. Further research was conducted by Abrams, McGrew, and Plotkin [1] to create a version of EigenTrust that satisfies constraint 1 by breaking these cycles probabilistically. However, the main difference between our proposed algorithm and the EigenTrust family lies in the subjective nature of the reputation scores assigned.

2.2 Appleseed

For the same reasons as EigenTrust, Appleseed [4] does not satisfy design constraint 3. However, it does provide subjective score using a similar initialization procedure to the one described in this algorithm. Much like our proposed algorithm, Appleseed can also be extended to include distrust.

2.3 TidalTrust

While TidalTrust [2] satisfies design constraint 3, it is difficult to evaluate constraint 1. TidalTrust uses shortest paths along directed edges to compute trust. Although this generally produces good results, it is possible that some node A placing a small amount of trust in a node B (as opposed to being unfamiliar with node B) will cause a decrease in the trust score of node B from a third observer. Additionally, this approach leaves out important context provided by longer paths.

3 Formalization of Criteria

To formalize our design criteria mathematically, we introduce the following definitions.

- Let $N(G)$ be the set of all nodes in G.
- Let $E(G)$ be the set of all edges in G.
- Let $F(G, A, B)$ be the trust score of node B from the perspective of node A where G is a weighted, directed graph such that $E(G)$ is a set of trust attestations and $A, B \in N(G)$. Each edge $E_0 \in E(G)$ has a source $S(E_0)$, target $T(E_0)$, and weight $W(G, S(E_0), T(E_0))$.

1. Non-Destructiveness. Let G' be a graph such that $N(G') = N(G)$ and $W(G, X, Y) = W(G', X, Y)$ for all $X, Y \in N(G)$ such that $X \neq M$ or $Y \neq B$. If $W(G', M, B) > W(G, M, B)$, then for all $A \in N(G)$, it must be true that $F(G', A, B) \geq F(G, A, B)$. If $W(G', M, B) < W(G, M, B)$, then for all $A \in N(G)$, it must be true that $F(G', A, B) \leq F(G, A, B)$.

2. Sybil-resistance and robustness towards malicious collectives. Let N_1, N_2, and N_3 be mutually exclusive sets of nodes in G such that $N_1 \cup N_2 \cup N_3 = N(G)$. Let N_1 represent the world outside of the collective. We let N_2 represent the creators of collective N_3, whose members are not trusted by members of N_1. Formally, $W(G, X_1, X_3) = 0$ for all $X_1 \in N_1$ and $X_3 \in N_3$. For all $A \in N_1$ and all $X_3 \in N_3$, it must be the case that $F(G, A, X_3) \leq F(G, A, X_2)$ for some X_2. That is, any outside node A must not assign a higher reputation to a collective member X_3 than it has assigned some collective creator X_2.

3. Balanced incentive structure. Let G' be a graph such that $N(G') = N(G)$ and $W(G, X, Y) = W(G', X, Y)$ for all $X, Y \in N(G)$ such that $X \neq M$ or $Y \neq B$. For all $A \in N(G)$, it must be true that $F(G', A, M) = F(G, A, M)$. We also require that $F(G', A, X) \geq F(G', A, M)$ for all X such that $F(G, A, X) \geq F(G, A, M)$ and $F(G', A, X) \leq F(G', A, M)$ for all X such that $F(G, A, X) \leq F(G, A, M)$.

4 Definition

4.1 Hyperparameters

The user-defined trust policy determines whether the set of weights originating from a given user is valid.
An example of such a trust policy is shown below. This policy allows each user to assign a weight of absolute value up to 0.8 to a maximum of 5 users.

Algorithm 1: Trust Policy

```
Data: $G, S$
$CTR = 0$ ;                                        /* Initialize counter*/
for $T \in N(G)$ do
    $CTR \leftarrow CTR + 1$ ;                     /* Increment counter*/
    if $W(G, S, T) > 0.8$ OR $CTR > 5$ then
        return False ;                              /* Policy failed*/
    end
end
return True ;                                       /* Policy passed*/
```

In addition to satisfying the trust policy, a graph G is valid only if

$$0 \leq W(G, S, T) \leq 1$$

for all $S, T \in N(G)$.

4.2 Notation

This algorithm takes place in a series of iterations. We maintain two mappings of nodes to numbers, Sc and In. We denote the values in these mappings corresponding to node A_0 as $Sc[A_0]$ and $In[A_0]$ respectively.

Let $K(N, In)$ be a function returning set of all nodes $n \in N$ such that $In[n] = 0$. Let $H(N, In, Sc)$ denote a function whose output is the set of all nodes $n \in K(N, In)$ such that $Sc[n] \geq Sc[m]$ for all nodes $m \in K(N, In)$. H returns the empty set if no such list exists.

Let $W(G, S, T)$ be the weight of the edge from S to T in graph G. Returns 0 if no such edge exists.

4.3 Pseudocode

We will define $F(G, A, B)$ using the following pseudocode.

Algorithm 2: Transitive Trust Algorithm

```
Data: $G, A, B$
$Sc[A] \leftarrow 1$ ;                                        /* Set A's reputation to 1.*/
for $X$ in $N(G)$ do
    if $X \neq A$ then
        $Sc[X] \leftarrow 0$ ;                   /* Set all other nodes' reputations to 0.*/
    end
    $In[X] \leftarrow 0$ ;                       /* None of the nodes have been inspected.*/
end
while $\exists X : In[X] = 0$ do
    $L \leftarrow H(N(G), In, Sc)$ ;        /* Let $L$ be the nodes of max accumulated score*/
    for $S$ in $L$ do
        $In[S] \leftarrow 1$ ;                                    /* Mark S as inspected.*/
        $Sc[S] \leftarrow MAX(Sc[S], 0)$ ;                  /* Reputation is non-negative*/
        for $T : S \neq T$ do
            if $In[T] = 0$ then
                $Sc[T] \leftarrow Sc[T] + (Sc[S] - Sc[T]) * W(G, S, T)$ ;     /* Pass score down*/
            end
        end
    end
end
return $Sc[B]$ ;                                          /* Return the score at B*/
```

5 Analysis of Criteria

We will now examine each of the criteria from section 2 and demonstrate that the proposed algorithm satisfies its mathematical definition.
For some $X, A, B \in N(G)$, we define $I[X][G][A][B]$ be the number i such that $X \in l$ on iteration i of the while loop in our algorithm. We denote $I[X][G][A][B]$ as I_X and $I[X][G'][A][B]$ as I'_X.

5.1 Non-Destructiveness

Let $Sc[G][i][X]$ be the value of $Sc[X]$ after i iterations of the while loop have been executed on graph G.

Let $In[G][i][X]$ be the value of $In[X]$ after i iterations of the while loop have been executed on graph G.

Let G' be a graph such that $N(G') = N(G)$ and $W(G, X, Y) = W(G', X, Y)$ for all $X, Y \in N(G)$ such that $X \neq M$ or $Y \neq B$. Suppose that $W(G', M, B) > W(G, M, B)$ and $F(G, A, B)$ and $F(G', A, B)$.

Lemma 1. *For all $i < I_B, I'_B$ and $X \neq B$, $Sc[G][i][X] = Sc[G'][i][X]$ for all X, $Sc[G][i][B] \leq Sc[G'][i][B]$ and $In[G][i][X] = In[G'][i][X]$ for all $X \neq B$.*

Proof. We induct on i to prove this.

If $i = 0$, $Sc[G][A][i] = Sc[G'][A][i] = 1$ and $Sc[G][i][X] = Sc[G'][i][X] = 0$ for all $X \neq A$. $In[G][i][X] = In[G'][i][X] = 0$ for all X.

Now, we make the following assumptions for some $i < I_B, I'_B$:

- $Sc[G][i-1][X] = Sc[G'][i-1][X]$ for all $X \neq B$

- $Sc[G][i-1][B] \leq Sc[G'][i-1][B]$
- $In[G][i-1][X] = In[G'][i-1][X]$ for all X

Let L be the list of nodes returned by $H(G, In, Sc)$ at the start of iteration i of the while loop computing $F(G, A, B)$. Let L' be the list of nodes returned by $L(G', In, Sc)$ at the start of iteration i of the while loop computing $F(G', A, B)$.

Let G_B be the subgraph of G such that $N(G_B) = N(G) \setminus \{B\}$ and $E(G_B) = \{E_B \in E(G) | S(E_B) \neq B, T(E_B) \neq B\}$. Notice that since $i < I_B, I'_B$, so B is not in l or l'. Therefore $L = H(G, In, Sc) = H(G_B, In, Sc)$ and $L' = H(G', In, Sc) = H(G'_B, In, Sc)$. Since $Sc[G][i-1][X] = Sc[G'][i-1][X]$, $In[G][i-1][X] = In[G'][i-1][X]$, and $G'_B = G_B$ for all $X \neq B$, $H(G_B, In, Sc) = H(G'_B, In, Sc)$. So $L = L'$, and thus $In[G][i][X] = In[G'][i][X]$ for all X.

First, note that $Sc[G][i][X] = Sc[G][i-1][X]$ for all X such that $In[G][i-1][X] = 1$ and $Sc[G'][i][X] = Sc[G'][i-1][X]$ for all X such that $In[G'][i][X] = 1$.
Now we compute $Sc[G][i][X]$ for all X such that $In[G][i][X] = 0$ according to our algorithm.

For all $S \in L$, $Sc[G][i][X] = MAX(Sc[G][i-1][X], 0)$ and $Sc[G'][i][X] = MAX(Sc[G][i-1][X], 0)$. Since $Sc[G][i-1][X] = Sc[G'][i-1][X]$, it follows that $Sc[G][i][X] = Sc[G'][i][X]$.
For all X such that $In[G][i][X] = 0$ and $X \notin L$, we compute $Sc[G][i][X]$ and $Sc[G'][i][X]$ as shown below. Let Sc_L be the score of any element of L.

$$Sc[G][i][X] = Sc[G][i-1][X] - (Sc_L - Sc[G][i-1][X]) \prod_{S \in L} (1 - W(G, S, X)) \tag{1}$$

$$Sc[G'][i][X] = Sc[G'][i-1][X] - (Sc_{L'} - Sc[G'][i-1][X]) \prod_{S \in L} (1 - W(G', S, X)) \tag{2}$$

Note that $L = L'$, so $Sc_L = Sc_{L'}$. If $X \neq B$, then $W(G, S, X) = W(G', S, X)$, so

$$\prod_{S \in L} (1 - W(G, S, X)) = \prod_{S \in L'} (1 - W(G', S, X)) \tag{3}$$

Hence $Sc[G][i][X] = Sc[G'][i][X]$ for all $X \neq B$.
If $X = B$, then $W(G, S, X) \leq W(G', S, X)$, so

$$\prod_{S \in L} (1 - W(G, S, X)) \geq \prod_{S \in L'} (1 - W(G', S, X)) \tag{4}$$

Since $Sc[G'][i-1][B] \geq Sc[G][i-1][B]$, substituting $X = B$ into equations (3) and (4) gives $Sc[G'][i][B] \geq Sc[G][i][B]$.

Now we have proven that for all $i < I_B, I'_B$, $Sc[G][i][X] = Sc[G'][i][X]$ for all $X \neq B$ and $Sc[G][i][B] \leq Sc[G'][i][B]$.

Next, we show that $I'_B \leq I_B$. Suppose for the sake of contradiction that $I'_B > I_B$. Then on iteration I_B in the computation of $F(G, A, B)$, $B \in L$ and on iteration I_B in the computation of $F(G', A, B)$, $B \notin L'$ and $In[G'][I_B][B] = 0$. This means that $Sc[G][I_B - 1][B] \geq Sc[G][I_B - 1][X]$ for all $X \notin B$. However, $Sc[G][I_B - 1][X] = Sc[G'][I_B - 1][X]$ for all $X \neq B$. So $Sc[G][I_B - 1][B] \geq Sc[G'][I_B - 1][X]$ for all $X \neq B$. Since $Sc[G'][i-1][B] \geq Sc[G][I_B - 1][B]$, $Sc[G'][I_B - 1][B] \geq Sc[G'][I_B - 1][X]$ for all $X \neq B$. Hence $B \in L'$, which is a contradiction.

Suppose $I_B = I'_B$. Then

$$\begin{aligned} F(G', A, B) &= MAX(Sc[G'][I'_B - 1][B], 0) \\ &\geq MAX(Sc[G][I_B - 1][B], 0) \\ &= F(G, A, B) \end{aligned}$$

Suppose $I'_B < I_B$. We know that $F(G', A, B) = Sc[G'][I'_B - 1][B]$ from the definition of our algorithm. Note that $Sc[G][I'_B - 1][X_0] > Sc[G][I'_B - 1][B]$ for some $X_0 \neq B$. Because $Sc[G'][I'_B - 1][B] \geq Sc[G'][I'_B - 1][X]$ and $Sc[G'][I'_B - 1][X] = Sc[G][I'_B - 1][X]$ for all X, we can conclude $Sc[G'][I'_B - 1][B] \geq Sc[G][I'_B - 1][X]$ for all X. $\square$

Since $F(G, A, B) \leq Sc[G][I'_B - 1][X_0]$, we make use of the inequality

$$Sc[G][I'_B - 1][X_0] \leq Sc[G'][I'_B - 1][B]$$

Because $F(G, A, B) \leq Sc[G][I'_B - 1][X_0] \leq Sc[G'][I'_B - 1][B] = F(G', A, B)$
By symmetry, we have $F(G', A, B) \leq F(G, A, B)$ when $W(G', M, B) < W(G, M, B)$.

5.2 Sybil-resistance and robustness towards malicious collectives

Let N_1,N_2,N_3 be defined as in section 2.2. Let $A \in N_1$, $X_3 \in N_3$ and let C be the first value of S in the computation of $F(G, A, B)$ such that $S \notin N_1$.

Note that for all X such that $I_X \geq I_C$, $F(G, A, X) \leq F(G, A, C)$. If $C \in N_2$, then $F(G, A, X_3) \leq F(G, A, C)$ for all $X_3 \in N_3$ because $I_{X_3} \geq I_C$ by definition of C.

If $C \in N_3$, it must be the case that $F(G, A, C) = 0$. This is because $F(G, A, C) = Sc[G][C][I_C - 1]$. Notice that $Sc[G][I_C - 1][C]$ must equal 0. This is because all previous values of S are in N_1, so only targets of edges E_0 such that $S(E_0) \in N_1$ are mutated after initialization. Since we are given that no such edge has a target in N_3, we know that $F(G, A, C) = Sc[G][I_C - 1][C] = 0$. By the same logic from the last paragraph, this tells us that $F(G, A, X) = 0$ for all $X \in N_2 \cup N_3$. Thus, our condition is satisfied in both possible cases.

5.3 Balanced Incentive Structure

Let G' be a graph such that $N(G') = N(G)$ and $W(G, X, Y) = W(G', X, Y)$ for all $X, Y \in N(G)$ such that $X \neq M$ or $Y \neq B$. Consider $F(G, A, B)$ and $F(G', A, B)$. By lemma 1, $Sc[G][i][X] = Sc[G'][i][X]$ for all $i < I_B$ and $X \neq B$.

Lemma 2. *For all $i \leq I_M$ and all $X \in N(G)$, $In[G][I_M][X] = In[G'][I_M][X]$ and $Sc[G][I_M - 1][X] = Sc[G'][I_M - 1][X]$.*

Proof. For some $i < I_M, I'_M$, assume $Sc[G][i - 1][X] = Sc[G'][i - 1][X]$ and $In[G][i - 1][X] = In[G'][i - 1][X]$ for all $X \in N(G)$.

Now, we compute $In[G][i][X]$ in all possible cases.
Case 1:
If $In[G][i - 1][X] = In[G'][i - 1][X] = 1$, then $In[G][i][X] = In[G'][i][X] = 1$.

Case 2: Since $N(G') = N(G)$, $In[G'][i - 1] = In[G][i - 1]$ and $Sc[G'][i - 1] = Sc[G][i - 1]$, it follows that

$$L = H(N(G), In[G][i - 1], Sc[G][i - 1]) = H(N(G'), In[G'][i - 1], Sc[G'][i - 1]) = L'$$

on iteration i. Since $L = L'$ on iteration i, it follows that $Sc_L = Sc_{L'}$. So for all X such that $X \in L$ on iteration i, $In[G][i][X] = In[G'][i][X] = 1$.

Case 3: If $In[G][i - 1][X] = In[G'][i - 1][X] = 0$, and $X \notin L$ on iteration i, then $X \notin L'$. So, $In[G][i][X] = In[G'][i][X] = 0$.

Therefore, $In[G][i][X] = In[G'][i][X]$.
Now, we compute $Sc[G][i][X]$ in all possible cases.

Case 1: If $In[G][i][X] = In[G'][i][X] = 1$, then $Sc[G][i][X] = Sc[G][i-1][X]$ and $Sc[G'][i][X] = Sc[G'][i-1][X]$. Therefore, since $Sc[G'][i-1][X] = Sc[G'][i-1][X]$, $Sc[G'][i][X] = Sc[G'][i][X]$.

Case 2: If $In[G][i][X] = In[G'][i][X] = 0$, then $Sc[G][i][X] = Sc[G][i-1][X]$ and $Sc[G'][i][X] = Sc[G'][i-1][X]$. Therefore, since $Sc[G'][i-1][X] = Sc[G][i-1][X]$, $Sc[G'][i][X] = Sc[G][i][X]$.
We compute $Sc[G][i][X]$ using equations (3) and (4).
Since $i < I'_M, I_M$, there is no $S \in L$ such that $S = M$. Therefore, $W(G, S, X) = W(G', S, X)$ for all $S \in l$. Therefore, since $Sc[G'][i-1][X] = Sc[G][i-1][X]$, $Sc[G'][i][X] = Sc[G][i][X]$.
By induction, $In[G'][I_M - 1][X] = In[G][I_M - 1][X]$ and $Sc[G'][I_M - 1][X] = Sc[G][I_M - 1][X]$ for all $X \in N(G)$. So, $l = l'$ on iteration I_M, and thus $In[G'][I_M][X] = In[G][I_M][X]$. □

Lemma 3. *For all $X \in N(G)$, if $I_X > I_M$ then $I'_X > I_M$.*

Proof. Suppose that $I_X > I_M$ and $I'_X \leq I_M$ then $In[G][I_M][X] = 0$ and $In[G'][I_M][X] = 1$. This is a contradiction to lemma 2.

□

If $I_M < I_B$, then $I_M < I'_B$ by lemma 3. So $Sc[G][I_M][M] = Sc[G'][I_M][M]$ by lemma 1. Since $F(G, A, M) = Sc[G][I_M][M]$ and $F(G', A, M) = Sc[G'][I_M][M]$, $F(G, A, M) = F(G', A, M)$.

If $I_M > I_B$, there exists no i such that $In[G][i][B] = 0$ and $In[G][i][M] = 1$, hence the value $W(G, B, M)$ is not computed in the algorithm, and the computation will yield the same result for any value of $W(G, B, M)$. Hence, $F(G, A, M) = F(G', A, M)$.

If $I_M = I_B$, there exists some i such that $M, B \in L(G, In, Sc)$. Since there exists exactly one i such that each node in $N(G)$ is in l, there exists no i such that $M \in L(G, In, Sc)$ and $B \notin l$.Hence the value $W(G, B, M)$ is not computed in the algorithm, so the computation will yield the same result for any value of $W(G, B, M)$. So, $F(G, A, M) = F(G', A, M)$. Thus we have proven the first part of our criteria.

Consider node X such that $F(G, A, X) \geq F(G, A, M)$, we know that $I_X \leq I_M$. By lemma 2, $Sc[G][X][I_X - 1] = Sc[G'][X][I_X - 1]$. By definition, $F(G, A, X) = MAX(Sc[G][X][I_X - 1], 0)$ and $F(G', A, X) = MAX(Sc[G'][X][I_X - 1], 0)$. So, $F(G', A, X) = F(G, A, X)$. Note that our previous proof yields $F(G', A, M) = F(G, A, M)$. So, it follows that $F(G', A, X) \geq F(G', A, M)$.

Consider node X such that $F(G, A, X) \leq F(G, A, M)$. Note that $I_X \geq I_M$, which implies that $I'_X \geq I_M$ by lemma 3. So $F(G', A, X) \leq F(G', A, M)$.

6 Enhancements and Modifications

A number of modifications can be made to the proposed algorithm to improve its robustness. One such modification is detailed below. For each of these modifications, it can be shown that the modified version satisfies all properties listed in section 2.

6.1 Incorporation of distrust

This version of the algorithm allows trusted nodes to propagate distrust using negative values for weights. Our new necessary condition for weights in a graph G is

$$|W(G, S, T)| < 1$$

for all $S, T \in N(G)$.

Algorithm 3: Transitive Trust Algorithm with Distrust

```
Data: $G, A, B$
$PSc[A] \leftarrow 1$ ;                                /* Set A's positive reputation to 1.*/
for $X$ in $N(G)$ do
    if $X \neq A$ then
        $PSc[X] \leftarrow 0$ ;            /* Set all other nodes' positive reputations to 0.*/
    end
    $NSc[X] \leftarrow 0$ ;                      /* Set all nodes' negative reputations to 0.*/
    $In[X] \leftarrow 0$ ;                          /* None of the nodes have been inspected.*/
end
while $\exists X : In[X] = 0$ do
    $L \leftarrow H(N(G), In, Sc)$ ;         /* Let $L$ be the nodes of max accumulated score*/
    for $S$ in $L$ do
        $In[S] \leftarrow 1$ ;                                        /* Mark S as inspected.*/
        $Sc[S] \leftarrow MAX(PSc[S] - NSc[S], 0)$ ;              /* Reputation is non-negative*/
        for $T : S \neq T$ do
            if $In[T] = 0$ then
                if $W(G, S, T) > 0$ AND $Sc[S] > PSc[T]$ then
                    $PSc[T] \leftarrow PSc[T] + (Sc[S] - PSc[T]) * W(G, S, T)$;
                end
                if $W(G, S, T) < 0$ AND $Sc[S] > NSc[T]$ then
                    $NSc[T] \leftarrow NSc[T] - (Sc[S] - NSc[T]) * W(G, S, T)$;
                end
            end
        end
    end
end
return $Sc[B]$ ;                                              /* Return the score at B*/
```

6.2 Disqualification of Nodes

On any iteration of the algorithm, if a node's score is less than a certain disqualification ceiling D, then it is disqualified from having a positive reputation (and by extension passing reputation to other nodes). This modification would leverage the loyalty of source node A to nodes that it sees with a higher reputation.

Algorithm 4: Transitive Trust Algorithm with Node Disqualification

```
Data: $G, A, B, D$
$PSc[A] \leftarrow 1$ ;                                  /* Set A's positive reputation to 1.*/
for $X$ in $N(G)$ do
    if $X \neq A$ then
        $PSc[X] \leftarrow 0$ ;          /* Set all other nodes' positive reputations to 0.*/
    end
    $NSc[X] \leftarrow 0$ ;                   /* Set all nodes' negative reputations to 0.*/
    $In[X] \leftarrow 0$ ;                       /* None of the nodes have been inspected.*/
end
while $\exists X : In[X] = 0$ do
    $L \leftarrow H(N(G), In, Sc)$ ;           /* Let $L$ be the nodes of max accumulated score*/
    for $S$ in $L$ do
        $In[S] \leftarrow 1$ ;                                     /* Mark S as inspected.*/
        $Sc[S] \leftarrow MAX(PSc[S] - NSc[S], 0)$ ;             /* Reputation is non-negative*/
    end
    for $C$ in $N(G)$ do
        $Dis[C] \leftarrow 0$ ;                            /* Assume C is not disqualified.*/
        if $PSc[C] - NSc[C] < D$ then
            $Dis[C] \leftarrow 1$ ;                                /* Mark C as disqualified.*/
        end
    end
    for $S$ in $L$ do
        for $T : S \neq T$ do
            if $In[T] = 0$ and $Dis[T] = 0$ then
                if $W(G, S, T) > 0$ AND $Sc[S] > PSc[T]$ then
                    $PSc[T] \leftarrow PSc[T] + (Sc[S] - PSc[T]) * W(G, S, T)$;
                end
                if $W(G, S, T) < 0$ AND $Sc[S] > NSc[T]$ then
                    $NSc[T] \leftarrow NSc[T] - (Sc[S] - NSc[T]) * W(G, S, T)$;
                end
            end
        end
    end
end
return $Sc[B]$ ;                                             /* Return the score at B*/
```

6.3 Telescoping Accumulation

An alternative model for the accumulation of trust is shown in the following algorithm. This method of accumulating trust amplifies the change in reputation caused by each one of n nodes of similar reputation attesting positively to a single node as n grows large. Note that this algorithm removes all restrictions on the value of $W(G, S, T)$.

Algorithm 5: Telescoping Transitive Trust Algorithm

```
Data: $G, A, B$
$Sc[A] \leftarrow 1$ ;                                   /* Set A's reputation to 1.*/
for $X$ in $N(G)$ do
    if $X \neq A$ then
        $Sc[X] \leftarrow 0$ ;                  /* Set all other nodes' reputations to 0.*/
    end
    $In[X] \leftarrow 0$ ;                      /* None of the nodes have been inspected.*/
end
while $\exists X : In[X] = 0$ do
    $L \leftarrow H(N(G), In, Sc)$ ;          /* Let $L$ be the nodes of max accumulated score*/
    for $S$ in $L$ do
        $In[S] \leftarrow 1$ ;                               /* Mark S as inspected.*/
        $Sc[S] \leftarrow MAX(Sc[S], 0)$ ;                   /* Reputation is non-negative*/
        for $T : S \neq T$ do
            if $In[T] = 0$ then
                $Sc[T] \leftarrow Sc[S] * \frac{Sc[T]+W(G,S,T)*(Sc[S]-Sc[T])}{Sc[S]+W(G,S,T)*(Sc[S]-Sc[T])}$ ;        /* Pass score down*/
            end
        end
    end
end
return $Sc[B]$ ;                                        /* Return the score at B*/
```

The same accumulation method can be applied to the variants discussed in 6.1 and 6.2 without compromising any of the properties shown in section 5. This can be done by replacing

$$PSc[T] \leftarrow PSc[T] + (Sc[S] - PSc[T]) * W(G, S, T)$$

with

$$PSc[T] \leftarrow Sc[S] * \frac{PSc[T] + W(G, S, T) * (Sc[S] - PSc[T])}{Sc[S] + W(G, S, T) * (Sc[S] - PSc[T])}$$

and replacing

$$NSc[T] \leftarrow NSc[T] + (Sc[S] - NSc[T]) * W(G, S, T)$$

with

$$NSc[T] \leftarrow Sc[S] * \frac{NSc[T] - W(G, S, T) * (Sc[S] - NSc[T])}{Sc[S] - W(G, S, T) * (Sc[S] - NSc[T])}$$

7 Time Complexity and Related Optimization

Let $\hat{N}(G)$ be the set of nodes X in graph G such that there exists a path from A to X along positively weighted edges. Let $\hat{E}(G)$ be the set of edges in graph G with their sources in $\hat{N}(G)$. Let $E = |\hat{E}(G)|$ and let $N = |\hat{N}(G)|$.

7.1 Discussion of Time Complexity

The big-O computational time complexity of computing $F(G, A, B)$ is at most $Elog(N)$. This is because, for all $X \in \hat{N}(G)$ the values X and $Sc[X]$ can be maintained in a max heap of size at most N.
These values can be queried and modified in at most $O(log(N))$ time, because they are members of a heap containing at most N elements. Since each node in G appears in L exactly once, all operations in the iteration over S are executed N times.
Letting a be the length of L, each computation of L is $O(alog(N))$. In total, the time complexity of these operations sums to $O(Nlog(N))$ because each node in G appears exactly once in L. Since each node is either the source or target of an edge, we know that $2E > N$. So, our operation is at most $O(Elog(N))$.
Notice that the operation in the iteration over T only mutates $Sc[T]$ when $W(G, S, T) \neq 0$ (i.e. there exists a directed edge from S to T). Therefore, we can instead iterate over edges E_0 such that $S(E_0) = S$. This approach results in our inner operation being executed exactly E times. Since the mutation of $Sc[T]$ is an $O(log(N))$ operation in this construct, we have bounded our total number of operations from above by a number on the order of $Elog(N)$.

7.2 Potential Optimization Leveraging Sparseness

Since each node represents a peer in a decentralized network, we expect our graph G to be sparse. To take advantage of the sparse nature of G, we can choose some constant k such that our algorithm iterates over at most k values of T for each value of S. This can also be implemented as a part of the trust policy.
Implementation of this limit further reduces the number of times that $Sc[T]$ is mutated to $k * N$, thus reducing overall time complexity to a maximum of $O(Nlog(N))$. However, this does come at some cost to the quality of results since we potentially ignore edges from any node A where $|\{E_0 : S(E_0) = A\}| > k$. The value of k determines where a modified version of the proposed algorithm lies with respect to quality and execution time. Such an optimization would effectively be an "n most-trusted (or least-trusted) entities" list for every peer in a network.

8 Output Interpretation

In some cases (especially when a trust policy requires that edges have a low weight), our algorithm outputs values that are near zero, but still positive. In these situations, it may be helpful to map reputation scores to outputs on a more useful scale. (E.g. $0.000167 \mapsto 0.1$ and $0.00545 \mapsto 0.2$)

This can be done using a variety of strictly increasing bijective functions from the interval $[0, 1]$ to itself. Some examples of such functions are

- x^n for constant n such that $0 < n < 1$
- $log(1 + nx)/log(1 + x)$ for constant $n > 1$

9 Conclusions and Implications

We have proposed a family of algorithms to compute reputation scores in an unmediated peer-to-peer environment while ensuring the integrity of the underlying signals created by each peer in the network. This can provide a basis for fully subjective digital experiences such as decentralized professional networks, review platforms, and file-sharing systems based on relative reputation. Given the sybil-resistant nature of these algorithms, such platforms can be created without imposing a cost to joining (a common stipulation of many P2P trust algorithms). Furthermore, our mechanism ensures that any non-malicious user allocates reputation in accordance with its own beliefs. This removal of selfish incentives makes our scheme easy to upgrade, as it ensures that the same set of signals (the weighted edges in our graph construct) can be used as the sole input to future algorithms.

10 Acknowledgement

Thanks to Steve Dakh for posing the problem of computing relative reputation as a function of a large public collection of digitally signed peer to peer attestations.

References

[1] Zoë Abrams, Robert McGrew, and Serge Plotkin. "A non-manipulable trust system based on EigenTrust". In: *ACM SIGecom Exchanges* 5.4 (July 2005), pp. 21–30. DOI: 10.1145/1120717.1120721.

[2] Jennifer Ann Goldbeck. "COMPUTING AND APPLYING TRUST IN WEB-BASED SOCIAL NETWORKS". PhD thesis. 2005.

[3] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. "The Eigentrust algorithm for reputation management in P2P networks". In: *Proceedings of the twelfth international conference on World Wide Web - WWW '03* (2003). DOI: 10.1145/775152.775242.

[4] Cai-Nicolas Ziegler and Georg Lausen. "Propagation models for trust and distrust in Social Networks". In: *Information Systems Frontiers* 7.4–5 (Dec. 2005), pp. 337–358. DOI: 10.1007/s10796-005-4807-3.