Edge Weight Prediction in Weighted Signed Networks

Srijan Kumar*, Francesca Spezzano†, V.S. Subrahmanian* and Christos Faloutsos‡
*University of Maryland, College Park, †Boise State University, ‡Carnegie Mellon University
*{srijan, vs}@cs.umd.edu, †francescaspezzano@boisestate.edu, ‡christos@cs.cmu.edu

***Abstract*—Weighted signed networks (WSNs) are networks in which edges are labeled with positive and negative weights. WSNs can capture like/dislike, trust/distrust, and other social relationships between people. In this paper, we consider the problem of predicting the weights of edges in such networks. We propose two novel measures of node behavior: the *goodness* of a node intuitively captures how much this node is liked/trusted by other nodes, while the *fairness* of a node captures how fair the node is in rating other nodes' likeability or trust level. We provide axioms that these two notions need to satisfy and show that past work does not meet these requirements for WSNs. We provide a mutually recursive definition of these two concepts and prove that they converge to a unique solution in linear time. We use the two measures to predict the edge weight in WSNs. Furthermore, we show that when compared against several individual algorithms from both the signed and unsigned social network literature, our fairness and goodness metrics almost always have the best predictive power. We then use these as features in different multiple regression models and show that we can predict edge weights on 2 Bitcoin WSNs, an Epinions WSN, 2 WSNs derived from Wikipedia, and a WSN derived from Twitter with more accurate results than past work. Moreover, fairness and goodness metrics form the most significant feature for prediction in most (but not all) cases.**

I. Introduction

A signed social network (SSN) is a network where edges may be labeled as being "positive" or "negative". For instance, if a vertex u dislikes a vertex v, there may be an edge with a "negative" edge label whereas if u likes v, the same edge labeled would be "positive". However, in the real-world, people may like or dislike one another with varying levels of intensity. Person A might dislike B a little bit, but dislike C a lot more. Or person A may trust B a little bit, but trust C a lot more. Or person A may disagree with B a little bit, but disagree with C a lot more. All of these concepts (liking, trusting, agreeing) are different and not necessarily symmetric, yet they all can be captured via (directed) weighted signed networks (WSNs).

A number of WSNs exist in the wild. For instance, we found two Bitcoin exchanges (OTC and Alpha), 2 Wikipedia networks, and an Epinions network which are explicit WSNs. In addition, we show how Twitter data can be viewed as a WSN. The meaning of positive and negative edges varies from network to network. However, WSNs, like any network, are incomplete. There may be like/dislike, trust/distrust or agree/disagree relations between people which we don't know about or are yet to form. For instance, person P1 may implicitly disagree with P2 on most things (e.g. if P1 mostly disagrees with P3 who mostly agrees with P2) even though they are not directly connected on Twitter. In this paper, we will study the problem of predicting the weights of edges in WSNs by examining data from all of the above networks. Prediction of such weights is significant for many reasons. For instance, we might wish to identify all people in a social network who might strongly agree with or support a particular topic (e.g. with a strength of over 0.8 on a weight scale that goes from -1 to +1) even if they have not tweeted about that topic, by seeing how much they like/agree or dislike/disagree with others who may have done so. As an example, a politician might look at such people as potential voters for him if he has strength in that particular topic of interest. In the same vein, we might wish to estimate how much a person P1 agrees/disagrees with people who are already positive or negative about a product (e.g. mobile phone carrier, airline) in order to quantify if they are more likely to be target customers of the product. Moreover, edge weight prediction may be useful to improve traditional tasks in signed networks such as node ranking [1], anomaly detection [2], [3], network analysis [4], [5], community detection [6], information diffusion [7], [8] and sentiment prediction [9], among others. Therefore, the prediction of edge weights in WSNs can be advantageous in various tasks, both when WSNs are explicit and implicit.

This paper focuses on the problem of *predicting the weight of edges in real-world WSN datasets for the first time*. A previous effort [10] studies WSNs without using naturally occurring WSNs. In this paper, we define two novel metrics of vertices that are unique to WSNs: *fairness*, to measure how fair is a vertex in assessing other vertices, and *goodness*, to measure how good do other vertices think this particular vertex is. We provide axioms that such fairness/goodness functions must satisfy and produce a specific definition of each that satisfy the axioms. We then develop a Fairness-Goodness Algorithm (FGA) that iteratively computes these two scores simultaneously and prove that this process is guaranteed to converge in linear time and moreover, that the fairness and goodness of a vertex is uniquely defined.

Based on this, we are the first to provide an algorithm to *predict edge-weights in WSNs*. By holding back data from 6 datasets, we are able to assess the accuracy of our methods with a series of four experiments:

(i) Leave-One-Out Prediction: When we use a common set of algorithms associated with SSNs (e.g. signed eigenvector centrality [11], status theory [5], [12], triadic balance [13],

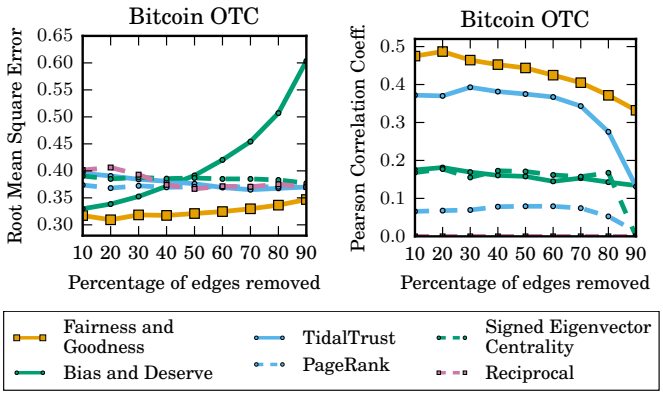


Fig. 1. The proposed metrics – Fairness and Goodness – perform the best, by having the lowest Root Mean Square Error and highest Pearson Correlation Coefficient, for predicting edge weights when N% of edge weights are removed from the network. The figure also shows that fairness and goodness metrics are robust, as the prediction performance is stable with increasing N.

PageRank [14], Signed-HITS [1], Bias and Deserve [15]) as well as the results of adaptations of past work on trust prediction such as TidalTrust [16], EigenTrust [17], and MDS [10], we show that *our fairness and goodness metrics are almost always the best when predicting the weight of an edge, one at a time.*

(ii) Leave $\mathcal{N}$% Out Prediction. When we remove $\mathcal{N}$% of the edges from the network and try to predict them from each of the individual features, we see that in all cases, fairness and goodness metrics are the best. We show the performance in case of one Bitcoin network in Figure 1. Here we see that *fairness and goodness has the lowest root mean square error and the highest Pearson correlation coefficient*, when comparing the predicted edge weight and the true edge weight, by varying the values of $\mathcal{N}$. Furthermore, the performance is stable, making the proposed metrics *robust to network sparsity* (*i.e.* when varying fraction of the network is not entirely visible to apps and users on Facebook, LinkedIn, etc.). Similar results for the other networks are discussed in detail in the Experiments section later (see Section VI).

(iii) Multiple Regression Prediction. When we use all of the aforementioned existing algorithms and proposed metrics for prediction in a supervised learning model using different regression models, then the performance improves. The root mean square error reduces to 0.22-0.32 and the Pearson Correlation Coefficient for all removed edges ranges from 0.46-0.81 and is significantly higher than any past predictive method. *Most importantly, in this ensemble, fairness and goodness metrics are the most important features in most of the networks.*

(iv) Multiple Regression with Leave $\mathcal{N}$% Out. Here, we again combine all algorithms to learn different regression models while removing $\mathcal{N}$% of the edges. This ensemble outperforms the past techniques and is also robust, due to the presence of the robust fairness and goodness metrics.

All the datasets and codes have been made available at `http://cs.umd.edu/~srijan/wsn`. The fairness and goodness metrics have also been implemented in the SNAP library [18] at `http://snap.stanford.edu/snap/`.

II. Related Work

Edge Sign Prediction in SSNs. Several papers have developed features and models to predict the sign of an edge by building upon balance and status theories [10], [19], [12], developing random-walk and trust-propagation algorithms [20], [21], and using social interaction information for prediction [22] (see [23] for a survey). Metric Multidimensional Scaling (MDS) [10] assigns an m-dimensional position to vertices in a SSN or WSN to minimize 'stress', based on extended balance theory. It then uses the metric distance between two vertices to predict the sign of an edge between them.

All of these papers predict edge sign in (unweighted) SSNs, while we predict the weight of an edge along with its sign. We compare with these past efforts and outperform them. For instance, we compare with balance theory, status theory, several random walk based centrality measures for WSNs and MDS algorithm, and show that FGA outperforms them predicting edge weights in WSNs (see Section VI for details).

Edge Weight Prediction in Social Networks. There is substantial work on predicting edge weights in ordinary (unsigned) social networks. Communication based features have been shown to be very important in quantifying the edge weights between users [24], [25], [26], [27], but since we only have the WSN without any communication information, we compare with non-communication based techniques. These include baselines such as reciprocal edge weight [24] and triadic balance and status measures [25], [28] (see Section VI for details). Two popular unsupervised algorithms are EigenTrust [17] and TidalTrust [16]. EigenTrust calculates a global value of trust for each vertex by finding the left eigenvector of a normalized trust matrix. TidalTrust calculates the trust from a source to a sink, by recursively propagating trust via the sink's predecessors till it reaches the source. A very recent work in this direction is trustingness and trustworthiness [29]. *However, these papers deal with edge weight prediction in unsigned social networks, while we look at the new problem of predicting edge weights in WSNs. So in this paper, we suitably adapt and compare with these techniques, and show that FGA outperforms them.*

III. Preliminaries

A *Weighted Signed Network (WSN)* is a directed, weighted graph $G = (V, E, W)$ where V is a set of users, $E \subseteq V \times V$ is a set of edges, and $W : E \to [-1, +1]$ is a mapping that assigns a value between -1 and +1 to each edge. $W(u, v)$ can be thought of as assigning a degree of "likes", "agrees" or "trust" score describing how much user u likes a user v. Notations used in the paper are described in Table I.

Given WSN $G = (V, E, W)$ and a node u we define the ego-in-network of u as the WSN ego-in$(u) = (V_u, E_u, W_u)$ where (i) $V_u = \{u\} \cup in(u)$, (ii) $E_u = \{(v, u)|v \in in(u)\}$, and (iii) $W_u(v, u) = W(v, u)$, $\forall (v, u) \in E_u$. We say that two nodes w_1 and w_2 have the identical ego-in-network iff $|in(w_1)| = |in(w_2)|$ and there exists a one-to-one mapping $h : in(w_1) \to in(w_2)$ s.t. $W_{w_1}(v, w_1) = W_{w_2}(h(v), w_2)$, $\forall v \in in(w_1)$. Similarly we can define the concept of ego-out-network of u, denoted by ego-out(u).

TABLE I
TABLE OF NOTATION USED IN THE PAPER.

A	Adjacency matrix of G=(V, E, W), *i.e.* A_{uv}=$W(u, v)$
A^+	Adjacency matrix of the positive graph only
A^-	Adjacency matrix of the negative graph only
$in(u)$	Set of nodes that preceed u
$in^+(u)$	Set of positive predecessors u
$in^-(u)$	Set of negative predecessors u
$out(u)$	Set of nodes that succeed u
$out^+(u)$	Set of positive successors u
$out^-(u)$	Set of negative successors u
$W^+_{in}(u)$	Total positive in-weight, $W^+_{in}(u) = \sum_{v \in in^+(u)} W(v, u)$
	$W^-_{in}(u)$, $W^+_{out}(u)$, and $W^-_{out}(u)$ are similarly defined.
PCC	Pearson correlation coefficient
p	p-value of the correlation

IV. ALGORITHM: "FAIRNESS" AND "GOODNESS"

In this section, we develop two measures (*fairness* and *goodness*) for each vertex in a WSN that will be used later for predicting edge weights.

The fairness of a vertex is a measure of how fair or reliable the vertex is in assigning ratings (like/dislike, agree/disagree, trust/distrust) to other vertices. Intuitively, a 'fair' or 'reliable' rater should give a user the rating that it deserves, while an 'unfair' one would deviate from that value. Hence, the ratings given by unfair raters should be given low importance, while ratings given by fair raters should be considered important. As an example, in real world networks such as the Bitcoin networks, scammers create multiple accounts to increase their own ratings and to reduce the ratings of benign users. To prevent this, scammers should be given a low fairness score.

On the other hand, the goodness of a vertex specifies how much other vertices like/dislike, agree/disagree, or trust/distrust that vertex and what its true quality is. This is the rating a totally fair vertex would give. Higher goodness implies the vertex is more trustworthy in the network. Hence, a 'good' or 'trustworthy' vertex would receive many high positive ratings from fair vertices, while a 'non-trustworthy' vertex would receive high negative ratings from fair vertices.

However, given a WSN, we don't know how fair and how good each vertex is. From the description above, it is clear that fairness and goodness metrics are dependent on each other. Therefore, we define how to assign fairness and goodness scores to each vertex in this section.

A. Prerequisites and Axioms

We start with two axioms that fairness and goodness metrics should satisfy.

*Axiom 1 (**Goodness axiom**):* Let u_1 and u_2 be two vertices having identical ego-in-networks, and let h be the 1-to-1 mapping between the two ego-in-networks. Then the Goodness axiom states that vertices with higher fairness have higher impact on the vertices they rate. Formally, if $f(v) = f(h(v))$ $\forall v \in in^-(u_1)$, and $f(v) \geq f(h(v))$ $\forall v \in in^+(u_1)$, then $g(u_1) \geq g(u_2)$. Conversely, if $f(v) = f(h(v))$ $\forall v \in in^+(u_1)$, and $f(v) \geq f(h(v))$ $\forall v \in in^-(u_1)$, then $g(u_1) \leq g(u_2)$.

*Axiom 2 (**Fairness axiom**):* Let u_1 and u_2 be two vertices having identical ego-out-networks, and let h be the 1-to-1 mapping between the two ego-out-networks. The Fairness axiom states that a vertex is more fair than another if it gives ratings closer to the rating deserved by the recipient. Formally, if $|W_{u_1}(u_1, k) - g(k)| \leq |W_{u_2}(u_2, h(k)) - g(h(k))|$ $\forall k \in out(u_1)$, then $f(u_1) \geq f(u_2)$.

TABLE II
COMPARISON OF FAIRNESS AND GOODNESS WITH TRADITIONAL METRICS THAT PROVIDE TWO SCORES TO EACH VERTEX. WE SEE THAT FAIRNESS AND GOODNESS HAS ALL THE DESIRED PROPERTIES.

Desired properties	Fairness and Goodness	Bias and Deserve	HITS
Satisfies Axioms 1 and 2	✓		
Scalable in $O(\|E\|)$	✓	✓	✓
Converges	✓	✓	✓
Robust to network sparsity	✓		

The Goodness axiom establishes dependence of goodness on fairness, and the Fairness axiom established the dependence of fairness on goodness.

Reasonable fairness and goodness measures should satisfy the above two axioms. In addition, the method to compute the fairness and goodness should have the following desirable properties: it should (a) **be scalable**, *i.e.* it should be linear in the number of edges in the network; (b) **guarantee a solution**, *i.e.* it should always converge and find a solution; (c) be **accurate**, *i.e.* it should be able to estimate the true quality of each vertex; and (d) be **robust**, *i.e.* it should perform well even when the network is not entirely visible.

B. Fairness and Goodness

We now provide two equations to compute the fairness and goodness metrics in a mutually recursive manner. We then show that they satisfy the axioms and have the desirable properties.

$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u) \times W(u, v) \quad (1)$$

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|W(u, v) - g(v)|}{R} \quad (2)$$

Fairness scores always lie in the $[0, 1]$ interval and goodness scores lie in the $[-1, 1]$ interval which is the domain of the edge weights in our case. The maximum possible difference between an edge weight and goodness score is the range of difference, $R = 2$.[1]

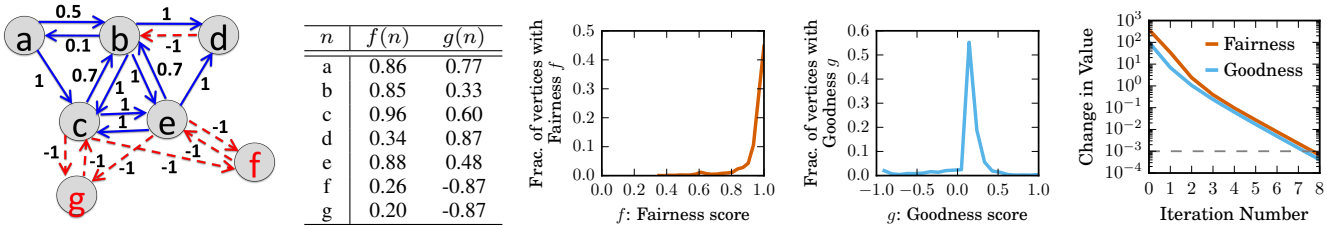
In the goodness formula for a vertex v, the incoming edge weights are weighted by the fairness of the vertices that are rating it, so that ratings by fair vertices are considered important. The average of these products over all predecessors yields the goodness of v. When calculating fairness of a vertex u, the smaller the difference between the actual edge weight and the goodness of the recipient, the more fair the vertex. Again, an average for all the ratings given by vertex u is used to calculate the fairness of u.

Proposition 1: Fairness and goodness measures satisfy both Axiom 1 and Axiom 2.

We prove the proposition formally in Appendix A.

Figure 3 shows the FGA algorithm to compute fairness and goodness scores for each vertex in the network. By default,

[1]If edge weights and goodness range over $[-\ell, \ell]$, then $R = 2\ell$.



n	$f(n)$	$g(n)$
a	0.86	0.77
b	0.85	0.33
c	0.96	0.60
d	0.34	0.87
e	0.88	0.48
f	0.26	-0.87
g	0.20	-0.87

(a) (b) (c) (d) (e)

Fig. 2. (a) An example network to explain fairness and goodness. (b) Fairness and goodness scores for each vertex in the example network. In the OTC network, the distribution of (c) fairness scores and (d) goodness scores. (e) Both fairness and goodness scores converge within 8 iterations in OTC network.

1: **Input**: A WSN $G = (V, E, W)$
2: **Output**: Fairness and Goodness scores for all vertices in V
3: Let $f^0(u) = 1$ and $g^0(u) = 1, \forall u \in V$
4: $t = -1$
5: **do**
6: $t = t + 1$
7: $g^{t+1}(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f^t(u) \times W(u, v), \forall v \in V$
8: $f^{t+1}(u) = 1 - \frac{1}{2|out(u)|} \sum_{v \in out(u)} |W(u, v) - g^{t+1}(v)|, \forall u \in V$
9: **while** $\sum_{u \in V} |f^{t+1}(u) - f^t(u)| > \epsilon$ or $\sum_{u \in V} |g^{t+1}(u) - g^t(u)| > \epsilon$
10: **Return** $f^{t+1}(u)$ and $g^{t+1}(u), \forall u \in V$

Fig. 3. Fairness and Goodness Algorithm (FGA)

the fairness and goodness scores of all vertices is set to 1 and 1, respectively (line 3). In line 7, the goodness score for each vertex is updated using the fairness scores from the previous iteration. In line 8, the fairness scores are updated using the newly updated goodness scores in the same iteration. For instance, after the first iteration, the goodness of all the vertices becomes their average in-degree. Both goodness and fairness scores are mutually recursive and are updated till both the scores converge (line 9). The algorithm converges when the change between fairness and goodness scores in consecutive iterations for all vertices is less than an error bound ϵ, which we set to 0.001. The scores of fairness and goodness from the last iteration are the final scores (line 10).

Example 1: Consider a sample weighted signed network in Figure 2(a), that very well reflects the real world scenario. Positive edges are shown in solid blue edges, negative ones are shown as red dotted edges. Vertices a, b, c and e are fair vertices, as their ratings are close to the other ratings that the recipients get. This makes their rating close to the goodness score of the vertex and this increases their fairness. Vertices d, f and g have lower fairness score, as their ratings deviate a lot from the goodness of their rating recipients. Looking at the goodness scores, vertices a, b, c, d and e are good, with the score of b being low positive as it receives edges of varying edge weights, most of which are positive. Vertices f and g have very negative goodness scores, as they get very negative ratings from very fair vertices. This example very well reflects the real world social networks, where majority of the vertices (a, b, c and e) are fair and good. Vertex d is interesting, as it is good but it is not fair. Vertices f and g are blatant trolls or fraudsters that have low fairness and very negative goodness scores. The scores of fairness and goodness for all the vertices is reported in Figure 2(b). □

Figure 2(c) and 2(d) show the fairness and goodness distribution for all the vertices in the OTC network. We see that most vertices have very high fairness (90% above 0.8 score; mean score = 0.94) meaning that most of the users are fair, but some vertices are not. For goodness, most vertices have low positive score (80% have score between 0 and 0.3), while a considerable fraction are considered 'not good' (14% have negative score, and 5% have goodness below -0.5). Similar observations hold for other networks.

C. Predicted Edge Weight by Fairness and Goodness

The weight of the edge (u, v) as predicted by the fairness and goodness is simply the the product $f(u) \times g(v)$. This way the predicted edge weight depends on both the fairness of the edge generator and the goodness of the edge recipient. We call this the FxG score of the edge.

D. Algorithm Analysis

In this part, we prove the convergence, uniqueness and the linear time complexity of the algorithm. For brevity, we state them here, and ***all the proofs are shown in Appendix A***.

*Theorem 1 (**Convergence Theorem**):* Let $f^t(u)$ be the fairness of node u after iteration t of the algorithm in Figure 3, and let $f^\infty(u)$ be its final score. Then, the error is bounded by $|f^\infty(u) - f^t(u)| \leq \frac{1}{2^t} \forall u \in V$. As t increases, the $f^t(u)$ converges to $f^\infty(u)$. Similarly, $|g^\infty(v) - g^t(v)| \leq \frac{1}{2^{t-1}} \forall v \in V$.

*Theorem 2 (**Uniqueness Theorem**):* FGA algorithm produces a unique solution of fairness and goodness scores upon convergence.

*Proposition 2 (**Linear Time Complexity of FGA**):* Let $G = (V, E, W)$ be a weighted signed network and $\epsilon > 0$. The time complexity of the algorithm to compute fairness and goodness (Figure 3) is $O(|E|)$.
We show the proof in Appendix B [30].

Figure 2(e) shows the change in value of fairness and goodness over each iteration for the case of OTC network. We observe FGA quickly terminates in 8 iterations.

Given that the formulation of Fairness and Goodness satisfies the axioms, converges to a unique score and has linear running time complexity, it satisfies the desirable properties. We will also show that this metric is accurate and robust in Section VI.

TABLE III
WEIGHTED SIGNED NETWORKS (WSNS) USED IN THE PAPER.

Network	Vertices	Edges	% Pos. Edges	Description of network edges
Bitcoin-OTC	5,881	35,592	89%	Degree of trust or distrust from Bitcoin user u to v
Bitcoin-Alpha	3,783	24,186	93%	Degree of trust or distrust from Bitcoin user u to v
Wikipedia RfA	9,654	104,554	83.7%	Degree of support or opposition in Wikipedia administrator elections by u to v
WikiSigned	341,646	5,625,022	83.02%	Degree of agreement or disagreement from Wikipedia editor u to v
Epinions	195,805	4,835,208	95.34%	Degree of trust or distrust from Epinions user u to v
Twitter	365,432	2,583,815	95.42%	Degree of positive or negative sentiment from Twitter user u to v

E. Comparison to traditional metrics

We now compare the Fairness and Goodness metrics with HITS [31] and Bias and Deserve [15]. Other centrality metrics, such as PageRank, Signed EigenVector Centrality, and others, assign only a single score to each vertex, and therefore are not directly comparable to the proposed fairness and goodness metrics.

Similarity and differences from HITS: Consider the special case when all $W(u,v) \in \{0,1\}$, *i.e.* when all edges are unweighted and unsigned. Then the formulation of fairness and goodness reduces to the following: for all edges (u,v), $W(u,v) = 1$ and since $g(v) < 1$, $|W(u,v) - g(v)| = |1 - g(v)| = 1 - g(v)$. So,

$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u)$$

and

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} 1 - g(v)$$
$$\Rightarrow f(u) = \frac{1}{|out(u)|} \sum_{v \in out(u)} g(v)$$

Note that in this special case of unweighted unsigned networks, fairness and goodness become very similar to the HITS formulation, with a difference in how the normalization is done. In HITS, normalization is done at the end of each iteration, so that the hub and authority vectors are unit vectors. In Equations 1 and 2, normalization is done for each vertex by averaging over its in- and out-edges, on the right hand side of the equation. This is done because when edge weights are negative in case of WSNs, the unit vector normalization is not valid. Since the original formulation of HITS is not defined for either SSNs or WSNs, it does not satisfy the axioms.

Similarity and differences from Bias and Deserve: A vertex u's bias (BIAS) reflects the expected weight of an outgoing edge to a random vertex, while its deserve (DES) reflects the expected weight of an incoming edge from an unbiased vertex [15]. As in the case of HITS, BIAS and DES are iteratively computed as:

$$DES^{t+1}(u) = \frac{1}{|in(u)|} \sum_{v \in in(u)} [W(v,u)(1 - X^t(v,u))]$$
$$BIAS^{t+1}(u) = \frac{1}{2|out(u)|} \sum_{v \in out(u)} [W(u,v) - DES^t(v)]$$

where $X^t(v,u) = \max(0, BIAS^t(v) \times W(v,u))$.

Proposition 3: Deserve satisfies Axiom 1 while bias does not satisfy Axiom 2.
We show the proof in Appendix B [30].

V. REAL-WORLD DATA: WEIGHTED SIGNED NETWORKS

Table III summarizes the statistics of the six real world weighted signed network (WSN) datasets used in this paper. All these networks are directed, i.e. each edge (u,v) means that user u is expressing an opinion about user v. Each edge is associated with a weight $W(u,v)$ representing the strength of opinion. The weights for all the networks are scaled to lie between -1 and 1. We explain the meaning of the sign and strength of edges in each of the different networks below. *The datasets have been released on the project webpage [30].*

Bitcoin networks. Bitcoin is a cryptocurrency that is used to trade anonymously over the web. Due to anonymity, there is counterparty risk [32], which has lead to the emergence of several exchanges where Bitcoin users rate the level of trust they have in other users. We created two datasets from two such exchanges - Bitcoin-OTC (OTC for short) and Bitcoin-Alpha[2] (Alpha for short). Both these exchanges allow users to rate others on a scale of -10 to +10 (excluding 0). According to OTC's guidelines, a rating of -10 should be given to fraudsters while at the other end of the spectrum, +10 means "you trust the person as you trust yourself". The other rating values have intermediate meanings. Therefore, these two exchanges explicitly yield WSNs. We scale the edge weights to be between -1 and 1.

Wikipedia RfA. Wikipedia Requests for Adminship (RfA) network [9] is a signed network among Wikipedia users where each edge (u,v) has a weight corresponding to the vote of user u (-1 for negative, 0 for neutral, and 1 for positive) towards user v to become an administrator. The dataset also contains a short explanation for each vote. We build a WSN from Wikipedia RfA network by weighting each edge with the intensity of the sentiment expressed in its explanation. We used an online VADER sentiment engine [33] to perform sentiment analysis on the explanations. It gives a positive sentiment score pos and a negative sentiment score neg to each explanation, where $-1 \leq pos, neg \leq +1$. We assign the sentiment score of the edge as $pos - neg$. The resulting network has weights in the range $[-1, 1]$.

WikiSigned. This WSN is between Wikipedia editors created in [34], where an edge from editor u to another editor v represents the trust or distrust u places in the edits made by v. The trust and distrust is calculated from the number of words from v's edits that u retains, replaces or deletes, and the number of edits u restores or reverts. The trust increases if the words are retained and edits restored, but distrust increases when words are replaced or deleted and edits reverted. The edge weight is then calculated as the difference between trust and distrust. The exact details are given in [34].

Epinions. We used the Extended Epinions dataset[3] to build a WSN as follows. In Epinions, each user u can rate the helpfulness of a review by user v on a 1-5 scale. There is an

[2]http://www.bitcoin-otc.com and http://www.btcalpha.com
[3]http://www.trustlet.org/extended_epinions.html

edge (u, v) from user u to user v if u expressed the helpfulness of at least one of v's reviews. We translated the helpfulness scores to the interval $[-1, +1]$ (a helpfulness score of 1, 2, 3, 4 and 5 are scaled to -1.0, -0.5, 0.0, 0.5 and 1.0, respectively), and set $w(u, v)$ to the average of the multiple helpfulness scores from u to v.

Twitter. We created a WSN from the Twitter India Election data reported in [35], [36] representing the average sentiment of a user towards another. There may be many tweets from u that mention v. For each of these tweets, we compute the sentiment of the tweet by using VADER, as was done for WikiRfA. The weight $W(u, v) \in [-1, 1]$ of edge from u to v is the average sentiment score of all tweets written by u that mention v.

More complex measures can be used to define the weight in the Twitter network, which could also include information about whether u retweeted or favorited a tweet by v and so forth. For simplicity, we use the sentiment of text as it systematically generates positive and negative sentiments. Moreover, the purpose of this paper is not to define how to create WSN, but to take a WSN as input for edge weight prediction. WSNs can also be created from Facebook, Tumblr, and other types of social media, as well as e-commerce sites such as Amazon, eBay (where the posts analyzed are reviews), possibly with combination of sentiment analysis. In this paper, we validate using the above defined six WSN datasets.

VI. Signed Edge Weight Prediction

In this section, we study the problem of predicting the weight of a signed directed edge (u, v) in WSNs. There is much work on predicting edge signs in signed networks — however, we predict not only the sign of an edge, but also its weight. *We are addressing this problem on WSNs for the first time.* By conducting experiments on 6 real-world described in Section V, we show that Fairness and Goodness outperforms existing algorithms, is the most important feature overall in multiple regression models, and is robust to sparsity.

Existing algorithms. As we discussed in the Related Work section, there has been significant work in developing metrics about signed social networks (SSNs), developing algorithms for edge sign prediction in SSNs and edge weight prediction in unsigned SNs. Since there is no work that addresses edge weight prediction in WSNs, we first show that adaptations of existing algorithms from these three fields are outperformed by our fairness and goodness metrics. However, we will use these past works as features in our multiple regression models, so all of these past works do provide value. The existing algorithms are:

• *Reciprocal*: The predicted weight for edge (u, v) is the same as that of the reciprocal edge (v, u) (0 if there is no reciprocal edge).

• *Triadic balance*: This is the average product of edge weights for all incomplete triads that the edge (u, v) is a part of. Incomplete triads are triads that would form involving edge (u, v) after it is created. This definition of triadic balance is derived directly from balance theory [13].

• *Triadic status*: This is the weight of the edge that is predicted by a simple extension of status theory [5] with edge weights using three rules. (i) In a transitive triad, the predicted weight of the transitive edge is the sum of the two non-transitive edge weights, and (ii) the predicted weight of a non-transitive edge is the difference of the transitive and the other non-transitive edge weights. (iii) In a cyclic triad, the weight of the missing edge is the negative of the sum of other two edge weights. Overall, the prediction by this baseline is the average predicted weight over all incomplete triads that an edge is a part of.

• *Status theory:* The prediction made by this measure is the difference between the status of vertex u and vertex v, defined as $\sigma(u) - \sigma(v)$. The status $\sigma(k)$ of vertex k is defined as $\sigma(k) = |W^+_{in}(k)| - |W^-_{in}(k)| + |W^-_{out}(k)| - |W^+_{out}(k)|$. Status [12] increases when receiving positive incoming edges and generating negative outgoing edges to other vertices, while decreases when receiving negative edges and generating outgoing positive edges. Difference in status measures how much 'higher' u's status is compared to v's. We extend the measure trivially to include weights instead of only signs.

• *PageRank:* The difference between weighted PageRank (wPR) value of vertex u and v, *i.e.* $wPR(u) - wPR(v)$, is the prediction made by this measure [14]. wPR is calculated in the unsigned version of the network as:

$$wPR(k) = \frac{1-\delta}{|V|} + \delta \sum_{z \in in(k)} \frac{wPR(z) \times W(z,k)}{|out(z)|}.$$

• *Signed Eigenvector Centrality:* The difference between signed eigenvector centrality (SEC) value of vertex u and v, *i.e.* $SEC(u) - SEC(v)$, is the prediction made by this measure. The SEC of vertices in a WSN with adjacency matrix A is the vector x that satisfies the equation $Ax = \lambda x$, where λ is the greatest eigenvalue [11].

• *Signed-HITS:* The prediction made by this model is the authority score $a(v)$ of vertex v computed by using a modified version of HITS for signed network, called Signed-HITS [1]. Signed-HITS iteratively computes the hub and authority scores separately on A^+ and A^-, using the equations:

$$h^+(u) = \sum_{v \in out^+(u)} a^+(v);\ a^+(u) = \sum_{v \in in^+(u)} h^+(v)$$

$$h^-(u) = \sum_{v \in out^-(u)} a^-(v);\ a^-(u) = \sum_{v \in in^-(u)} h^-(v)$$

and assigns, after convergence, the authority score $a(u) = a^+(u) - a^-(u)$ and hub score $h(u) = h^+(u) - h^-(u)$ to each vertex u. Again, we trivially extend the formula to include edge weights.

• *Bias and Deserve:* This is the measure proposed in [15], as discussed in Section IV. The deserve value $DES(v)$ of the vertex v is the prediction made by this algorithm.

• *TidalTrust:* We adapt the popular trust prediction model TidalTrust [16] to work on WSNs. We calculate the predicted trust values from u to v separately in the positive and negative sub-networks, and then take the difference between them as the feature for prediction.

TABLE IV

PERFORMANCE OF ALL COMPETITORS FOR SIGNED EDGE WEIGHT PREDICTION IN LEAVE-ONE-OUT SETTING. EACH CELL REPORTS $(RMSE, PCC)$. LOWER RMSE AND HIGHER PCC ARE DESIRABLE. FAIRNESS AND GOODNESS PERFORMS THE BEST.

Method	OTC	Alpha	WikiRfA	Twitter	Epinions	WikiSigned
Existing algorithms						
Reciprocal	(0.32, 0.46)	(**0.27**, **0.47**)	(0.35, 0.04)	(0.94, -0.11)	(0.71, 0.24)	(0.56, 0.04)
Triadic Balance	(0.57, 0.38)	(0.57, 0.25)	(0.75, 0.20)	(0.36, 0.22)	(0.41, 0.49)	(0.71, 0.62)
Triadic Status	(0.65, 0.22)	(0.65, 0.26)	(0.69, 0.20)	(0.44, 0.26)	(0.97, 0.12)	(0.68, **0.65**)
Status Theory	(1.03, 0.16)	(0.97, -0.12)	(1.19, 0.04)	(1.90, 0.00)	(1.55, -0.03)	(1.33, -0.25)
PageRank	(0.37, 0.07)	(0.32, -0.02)	(0.34, -0.04)	(0.94, 0.00)	(0.87, -0.16)	(0.55, 0.19)
Signed Eigenvector Centrality	(0.37, 0.17)	(0.32, 0.23)	(0.34, 0.11)	(0.98, -0.09)	(0.87, -0.20)	(0.56, -0.01)
Signed-HITS	(0.37, 0.34)	(0.32, 0.25)	(0.35, 0.22)	(0.93, 0.29)	(0.86, 0.38)	(0.57, 0.17)
Bias and Deserve	(0.36, 0.32)	(0.31, 0.24)	(**0.23**, 0.44)	(**0.30**, 0.37)	(0.38, 0.53)	(0.47, 0.31)
TidalTrust	(0.39, 0.38)	(0.36, 0.22)	(0.31, 0.24)	(0.33, 0.39)	(0.41, 0.44)	(0.58, 0.44)
EigenTrust	(0.37, 0.01)	(0.33, 0.00)	(0.35, 0.00)	(0.94, 0.00)	(0.87, 0.00)	(0.55, 0.00)
MDS	(0.35, 0.41)	(0.31, 0.32)	(0.33, 0.28)	(0.41, 0.27)	(0.42, 0.39)	(0.58, 0.36)
Proposed algorithm						
Goodness	(0.32, 0.48)	(**0.27**, 0.40)	**(0.23, 0.46)**	**(0.30, 0.41)**	(**0.32**, 0.64)	(0.47, 0.31)
Fairness × Goodness (FxG)	**(0.31, 0.49)**	(**0.27**, 0.41)	(0.24, 0.45)	(0.36, **0.41**)	(0.36, **0.65**)	(**0.46**, 0.40)

TABLE V

SUPERVISED CLASSIFICATION USING PREDICTION BY ALL ALGORITHMS DESCRIBED ABOVE AS FEATURES.

Method	OTC	Alpha	WikiRfA	Twitter	Epinions	WikiSigned
Linear Regression, LR(FxG+)	(**0.26**, 0.66)	**(0.22, 0.62)**	**(0.22, 0.50)**	**(0.25, 0.46)**	**(0.29, 0.67)**	(0.32, 0.75)
Ridge Regression, RR(FxG+)	(**0.26, 0.67**)	(0.23, 0.62)	**(0.22, 0.50)**	(0.26, 0.43)	(**0.29**, 0.65)	**(0.29, 0.81)**
Lasso, $Lasso$(FxG+)	(0.27, 0.66)	(0.23, 0.61)	(0.23, 0.47)	(0.26, 0.45)	(0.30, 0.63)	(0.30, 0.77)
ElasticNet, EN(FxG+)	(0.27, 0.65)	(0.24, 0.60)	(0.23, 0.47)	(0.26, 0.43)	(**0.29**, 0.64)	(**0.29**, 0.77)

TABLE VI

FEATURE IMPORTANCE USING UNIVARIATE LINEAR REGRESSION TEST. THE FEATURES ARE SORTED ACCORDING TO DECREASING ORDER OF IMPORTANCE FOR EACH NETWORK INDIVIDUALLY. ALL P-VALUES ARE BELOW 0.0001.

OTC	Alpha	WikiRfA	Twitter	Epinions	WikiSigned
Fairness × Goodness	Reciprocal	*Goodness*	*Fairness × Goodness*	*Fairness × Goodness*	Triadic Status
Goodness	*Fairness × Goodness*	*Fairness × Goodness*	*Goodness*	*Goodness*	Triadic Balance
Reciprocal	*Goodness*	Bias and Deserve	TidalTrust	Bias and Deserve	TidalTrust
MDS	MDS	MDS	Bias and Deserve	Triadic Balance	*Fairness × Goodness*
TidalTrust	Signed-HITS	TidalTrust	Signed-HITS	TidalTrust	MDS

• *EigenTrust:* EigenTrust, ET, is another popular trust prediction algorithm [17]. The prediction is the difference in the predicted trust of the two vertices involved, $ET(u) - ET(v)$.

• *MDS:* This is a recent algorithm proposed to work for trust prediction on SSNs [10](details of the algorithm are provided in Section II). The convergence model in MDS can only accept edge weights in {-2,-1,+1,+2}, so we convert the weights of all networks by scaling and rounding off to these values. The edge weight is predicted as the distance between the embeddings of the two vertices of the edge, as generated by MDS.

Prediction by Fairness and Goodness: Lastly, we have the metrics we propose in this paper. We use the goodness score $g(v)$ of the vertex v, and the FxG score, $f(u) \times g(v)$ as two separate predictions made using Fairness and Goodness.

Evaluation metrics. The performance of edge weight prediction is measured using two standard metrics – Root Mean Square Error, $RMSE$, between the actual edge weight and the predicted weight ($RMSE$ lies in the range of 0 to 2, since edge weights are between -1 and +1; 0 is the best, 2 is the worst), and Pearson Correlation Coefficient, PCC, (range is -1 to 1; random is 0). Both these measures characterize different aspects of the prediction – $RMSE$ quantifies how close the prediction is to the true values on average, and PCC measures the relative trend between the two.

Results. We perform a series of experiments to evaluate the performance of various features for edge weight prediction.

Experiment 1: Leave One-Out Edge Weight Prediction: We conduct a Leave-One-Out cross validation experiment, which captures the scenario where a WSN exists and we want to find the weight of a non-existent edge. In this experiment, one test edge e_t is removed at a time and the task is to predict its edge weight, given the rest of the network. Each algorithm makes its prediction based on the remaining network. The $RMSE$ and PCC values are calculated for each e_t, and averaged. This is an entirely *unsupervised* approach. Table IV shows the average $RMSE$ and PCC of each algorithm. **We observe that FxG is almost always the best method across all 6 datasets.**

Experiment 2: Multiple Regression Models for Edge Weight Prediction: This experiment builds a supervised learning model by using each of the algorithms in Experiment 1 to generate a feature. Each feature is the edge weight prediction by that algorithm. The procedure used is the following: the test edge e_t whose weight is to be predicted is removed from the network. From the remaining edges, a second edge e_s is removed at a time to generate the prediction using all the algorithms. This step is done multiple times to generate training examples, with the features being the predictions and the label being the edge weight of e_s. The training is done using 4 regression models (Linear Regression, Ridge Regression, Lasso, and ElasticNet), and the learned model is used to predict the edge weight of e_t. $RMSE$ and PCC values are calculated for this prediction. This entire process

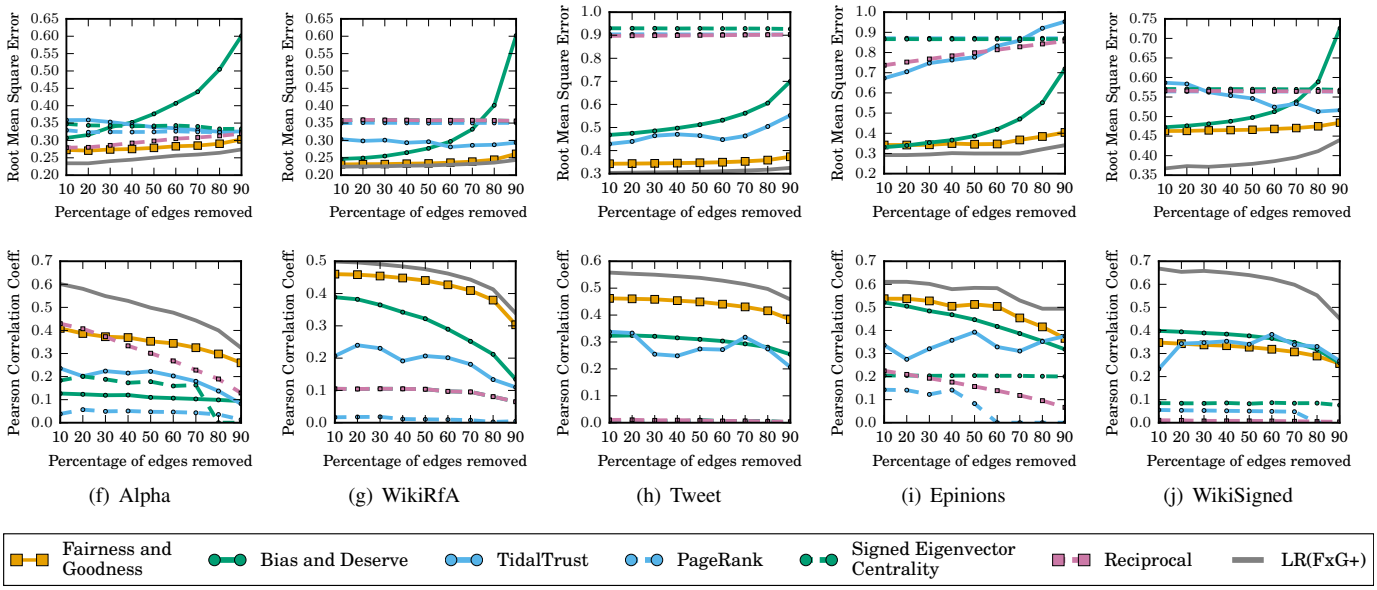


Fig. 4. Fairness and Goodness metrics are highly robust to network sparsity. This plot shows the variation of RMSE (top) and PCC (bottom) for five networks (the plot for OTC is in Figure 1), when a random subset of edges are removed from each network. Some measures are removed to enhance visibility. We observe that over all networks, the fairness and goodness metrics perform the best individually as the percentage of edges removed increases. Linear Regression performs the best overall, with fairness and goodness as the most important feature.

is repeated multiple times by sampling different e_t and the performance metrics are averaged. The regression models are named LR(FxG+) for Linear Regression, RR(FxG+) for Ridge Regression, and so on, as it takes a feature vector that has Fairness and Goodness predictions as features (and they are the most important features as seen later).

Table V shows that predictive power improves significantly for all networks, both in terms of $RMSE$ and PCC, compared to the unsupervised technique explained in the previous experiment. We are able to predict edge weights with a PCC ranging in 0.46-0.81, and the $RMSE$ ranging in 0.22-0.32 depending on the network studied. This is important, as practitioners can directly take this model and apply it in real world. In general, linear regression seems to give the best results.

Table VI shows the 5 most important features (in order) for each network obtained using the `f_regression` method implemented in Scikit Learn for identifying the most important features in regression models. The results are all statistically significant ($p < 0.0001$). **We observe that FxG is ranked #1 in 3 of the 6 networks, #2 in 2, and #4 in one. The Goodness measure alone does slightly worse. No other measure comes close.**

Experiment 3: Effect of removing $\mathcal{N}\%$ of edges: In this experiment, we remove a larger fraction of edges by randomly selecting $\mathcal{N}\%$ of edges, compute features for the remaining edges and then predict the weight of the removed edges (similar to the process done in Experiment 1, but with more edges removed as test edges). We vary the value of $\mathcal{N}$, from 10 to 90% in steps of 10%, to observe the effect of fraction of missing edges from the network. For each $\mathcal{N}$, we randomly generate 100 networks and take the mean of the results. Figure 1 shows the $RMSE$ and PCC of each feature in predicting the weight of the missing edges in the OTC network, and Figure 4 shows them for the remaining five networks. (The curve labeled linear regression will be discussed in Experiment 4). We only show the important curves to enhance visibility. **We see that of all the features, the one that generates the best prediction is the FxG feature.** We note that the confidence interval of performance is very small in each case, so we don't show it in the figure for better clarity.

Moreover, this figure shows that FxG is very robust to partial visibility of the network. For instance, part of a network may be invisible to an application due to privacy constraints (*e.g.* on Facebook and LinkedIn, we can only see part of the network). In fact, as more edges are removed from the network, FxG outperforms all other features used. Interestingly, we observe that features that performed well in Experiment 1 degrade more than FxG. For example, consider the "Reciprocal" feature in the Alpha network - it degrades rapidly as visibility of the network is reduced. Likewise, Bias and Deserve, which is the closest competitor in most networks, is not robust at all in terms of $RMSE$, as its performance degrades quickly and by a large margin.

Experiment 4: Effect of removing $\mathcal{N}\%$ of edges, with Multiple Regression: As in Experiment 2, we learn a supervised learning model by training on networks with $\mathcal{N}\%$ edges removed, and testing to predict the edge weight of the remaining edges. This is shown as the gray lines in Figure 4, which again shows that by training on the given data, the performance for edge weight prediction in WSNs improves (this line is omitted from Figure 1 for OTC network for clarity, but the same is observed there as well). The $RMSE$ values are lower and the PCC values are higher when the regression model is used. This again implies large practical application to improve

WSN based services. The prediction is done using a linear regression model and is called LR(FxG+) as before because Fairness and Goodness metrics are the most important features. This is especially true when higher percentage of edges are removed from the networks. As previously, the importance of features was calculated using `f_regression` method. Other regression models also perform similarly.

VII. Conclusion

Our paper is the first to show how to predict edge weights in weighted signed networks (WSNs) and has the following contributions:

• **Novel metrics:** We proposed two vertex based metrics called fairness and goodness to assess the reliability of a node in rating others, and to assess how much the node is liked/trusted by other nodes, respectively.

• **Convergence and uniqueness:** We show that fairness and goodness converge to a unique value in time linear to the size of the network.

• **Effectiveness in Signed Edge Weight Prediction:** We show that fairness and goodness can be used to calculate unknown weights in WSNs with higher precision than previous techniques. In conjunction with other features defined for WSNs, we show that our prediction engine is able to outperform past methods (suitably enhanced).

• **Robustness.** We also show the robustness of the engine by varying the size of the networks by showing that it performs the best across all these networks.

All datasets and codes have been made available at [30]. The codes are also open-sourced in the SNAP library [18].

Acknowledgement

Part of this work was funded by the US Army Research Office under Grant Number W911NF1610342. We would like to thank Andrew Miller and the anonymous reviewers for useful comments.

References

[1] M. Shahriari and M. Jalili, "Ranking nodes in signed social networks," *Social Network Analysis and Mining*, vol. 4, no. 1, 2014.
[2] S. Kumar, F. Spezzano, and V. Subrahmanian, "Accurately detecting trolls in slashdot zoo via decluttering," in *ASONAM*, 2014.
[3] Z. Wu, C. C. Aggarwal, and J. Sun, "The troll-trust model for ranking in signed networks," in *WSDM*, 2016.
[4] S. Kumar, "Structure and dynamics of signed citation networks," in *WWW Companion*, 2016.
[5] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *SIGCHI*, 2010.
[6] V. A. Traag and J. Bruggeman, "Community detection in networks with positive and negative links," *Physical Review E*, vol. 80, no. 3, 2009.
[7] M. Shafaei and M. Jalili, "Community structure and information cascade in signed networks," *New Generation Computing*, vol. 32, no. 3-4, pp. 257–269, 2014.
[8] D. Li, Z.-M. Xu, N. Chakraborty, A. Gupta, K. Sycara, and S. Li, "Polarity related influence maximization in signed social networks," *PloS one*, vol. 9, no. 7, 2014.
[9] R. West, H. S. Paskov, J. Leskovec, and C. Potts, "Exploiting social network structure for person-to-person sentiment analysis," *TACL*, vol. 2, pp. 297–310, 2014.
[10] Y. Qian and S. Adali, "Foundations of trust and distrust in networks: Extended structural balance theory," *TWEB*, vol. 8, no. 3, p. 13, 2014.
[11] P. Bonacich and P. Lloyd, "Calculating status with negative relations," *Social Networks*, vol. 26, no. 4, pp. 331 – 338, 2004.
[12] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *WWW*, 2010.
[13] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory." *Psychological review*, vol. 63, no. 5, p. 277, 1956.
[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, no. 1-7, pp. 107–117, 1998.
[15] A. Mishra and A. Bhattacharya, "Finding the bias and prestige of nodes in networks based on trust scores," in *WWW*, 2011.
[16] Y. Katz and J. Golbeck, "Social network-based trust in prioritized default logic," in *AAAI*, 2006.
[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW*, 2003.
[18] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, p. 1, 2016.
[19] B. Huang, A. Kimmig, L. Getoor, and J. Golbeck, "A flexible framework for probabilistic models of social trust," in *SBP*, 2013.
[20] T. DuBois, J. Golbeck, and A. Srinivasan, "Predicting trust and distrust in social networks," in *PASSAT/SocialCom*, 2011.
[21] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *WWW*, 2004.
[22] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang, "Friend or frenemy?: predicting signed ties in social networks," in *SIGIR*, 2012.
[23] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Computing Surveys*, vol. 49, no. 3, pp. 1–37, 2016.
[24] E. Gilbert, "Predicting tie strength in a new medium," in *CSCW*, 2012.
[25] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in *SIGCHI*, 2009.
[26] I. Kahanda and J. Neville, "Using transactional information to predict link strength in online social networks." in *ICWSM*, 2009.
[27] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *WWW*, 2010.
[28] S. Sintos and P. Tsaparas, "Using strong triadic closure to characterize ties in social networks," in *KDD*, 2014.
[29] A. Roy, J. Srivastava, and J. Huh, "Trustingness & trustworthiness: A pair of complementary trust measures in a social network," in *ASONAM*, 2016.
[30] "Project webpage," http://cs.umd.edu/~srijan/wsn/.
[31] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632.
[32] T. Moore and N. Christin, "Beware the middleman: Empirical analysis of bitcoin-exchange risk," in *Financial cryptography (FC)*, 2013.
[33] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.
[34] S. Maniu, B. Cautis, and T. Abdessalem, "Building a signed network from interactions in wikipedia," in *DBSocial*, 2011.
[35] J. P. Dickerson, V. Kagan, and V. Subrahmanian, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?" in *ASONAM*, 2014.
[36] V. Kagan, A. Stevens, and V. Subrahmanian, "Using twitter sentiment to forecast the 2013 pakistani election and the 2014 indian election," *IEEE Intelligent Systems*, vol. 30, no. 1, pp. 2–5, 2015.

Appendix A
Fairness and Goodness proofs

Proof for Proposition 1: Fairness and Goodness satisfy Axiom 1 and Axiom 2.

Proof: Let us start by proving that Axiom 1 is satisfied by goodness. Let u_1 and u_2 be two vertices having the identical ego-in-network, and let h be the 1-to-1 mapping between the two ego-in-networks. From definition of goodness, we have

$$\begin{aligned} & g(u_1) - g(u_2) = \\ & = \tfrac{1}{|in(u_1)|} \textstyle\sum_{v' \in in(u_1)} f(v') \cdot W_{u_1}(v', u_1) + \\ & \qquad - \tfrac{1}{|in(u_2)|} \textstyle\sum_{v'' \in in(u_2)} f(v'') \cdot W_{u_2}(v'', u_2) \\ & = \tfrac{1}{|in(u_1)|} \Big(\textstyle\sum_{v' \in in^+(u_1)} (f(v') - f(h(v'))) \cdot W_{u_1}(v', u_1) + \\ & \qquad + \textstyle\sum_{v' \in in^-(u_1)} (f(v') - f(h(v'))) \cdot W_{u_1}(v', u_1) \Big) \end{aligned}$$

If $f(v) = f(h(v))\ \forall v \in in^-(u_1)$, and $f(v) \geq f(h(v))\ \forall v \in in^+(u_1)$, it follows that

$$g(u_1) - g(u_2) = \tfrac{1}{|in(u_1)|} \textstyle\sum_{v' \in in^+(u_1)} (f(v') - f(h(v')) \cdot W_{u_1}(u, v) + 0$$

Since $f(v) \geq f(h(v))\ \forall v \in in^+(u_1)$, we have that $\sum_{v' \in in^+(u_1)}(f(v') - f(h(v')) \cdot W_{u_1}(u, v) \geq 0$, and, consequently $g(u_1) - g(u_2) \geq 0$ that implies $g(u_1) \geq g(u_2)$.

A symmetric proof can be done to show that, if $f(v) = f(h(v))\ \forall v \in in^+(u_1)$, and $f(v) \geq f(h(v))\ \forall v \in in^-(u_1)$, then $g(u_1) \leq g(u_2)$.

Let us now prove that Axiom 2 is satisfied by fairness. Let u_1 and u_2 be two vertices having the identical ego-out-network, and let h' be the 1-to-1 mapping between the two ego-out-networks. From definition of fairness, we have that

$$f(u_1) - f(u_2) = -\frac{1}{|out(u_1)|}\sum_{v \in out(u_1)} \frac{|W_{u_1}(u_1,v) - g(v)|}{R} + \frac{1}{|out(u_2)|}\sum_{v' \in out(u_2)} \frac{|W_{u_2}(u_2,v') - g(v')|}{R} =$$
$$\frac{1}{|out(u_1)|R}\sum_{v \in out(u_1)} |W_{u_2}(u_2, h'(v)) - g(h'(v))| - |W_{u_1}(u_1, v) - g(v)|$$

Since $|W_{u_1}(u_1, v) - g(v)| \leq |W_{u_2}(u_2, h'(v)) - g(h'(v))|\ \forall v \in out(u_1)$, it follows that

$$\sum_{v \in out(u_1)} |W_{u_2}(u_2, h'(v)) - g(h'(v))| - |W_{u_1}(u_1, v) - g(v)| \geq 0$$

and, consequently $f(u_1) - f(u_2) \geq 0$ that implies $f(u_1) \geq f(u_2)$. ∎

Proof for Theorem 1: Convergence of Fairness and Goodness scores

Proof: Let us prove convergence of fairness scores.

The theorem is proven by induction. Given the definition of fairness and goodness, the fairness scores $f^\infty(u)$ and $f^{t+1}(u)$ can be recursively written as

$$f^\infty(u) = 1 - \frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^\infty(k)}{|in(v)|}|$$

$$f^{t+1}(u) = 1 - \frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^t(k)}{|in(v)|}|$$

Base case: $t = 1$. In the first iteration, we have

$$|f^\infty(u) - f^1(u)| =$$
$$|\Big(\frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^0(k)}{|in(v)|}|\Big) -$$
$$\Big(\frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^\infty(k)}{|in(v)|}|\Big)|$$
$$\implies |f^\infty(u) - f^1(u)| =$$
$$|\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(|W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^0(k)}{|in(v)|}| -$$
$$|W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f^\infty(k)}{|in(v)|}|\Big)|$$

As $|x| - |y| \leq |x - y|$, applying it to the two terms within the summation, we have

$$\implies |f^\infty(u) - f^1(u)| \leq$$
$$|\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(|\frac{\sum_{k \in in(v)} W(k,v) f^\infty(k)}{|in(v)|} - \frac{\sum_{k \in in(v)} W(k,v) f^0(k)}{|in(v)|}|\Big)|$$
$$= \frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(|\frac{\sum_{k \in in(v)} W(k,v)(f^\infty(k) - f^0(k))}{|in(v)|}|\Big)$$

As $|x + y| \leq |x| + |y|$, we get

$$|f^\infty(u) - f^1(u)| \leq \frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} |W(k,v)(f^\infty(k) - f^0(k))|}{|in(v)|}$$

At this point, as $|x \cdot y| \leq |x| \cdot |y|$, we have

$$|W(k,v)(f^\infty(k) - f^0(k))| \leq |W(k,v)| \cdot |f^\infty(k) - f^0(k)|$$

We observe that $|W(k,v)| \leq 1$ and, since $f(k) \in [0,1]$, $|f^\infty(k) - f^0(k)| \leq 1$. Then, $|W(k,v)| \cdot |f^\infty(k) - f^0(k)| \leq 1$.

Thus,

$$|f^\infty(u) - f^1(u)| \leq$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} |W(k,v)| \cdot |(f^\infty(k) - f^0(k))|}{|in(v)|} \leq$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} 1}{|in(v)|} = \frac{1}{2|out(u)|}\sum_{v \in out(u)} 1 = \frac{1}{2}$$
$$\implies |f^\infty(u) - f^1(u)| \leq \frac{1}{2}$$

Induction step. Let us assume by hypothesis that $|f^\infty(u) - f^t(u)| \leq \frac{1}{2^t}$. Then,

$$|f^\infty(u) - f^{t+1}(u)| \leq$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(\frac{\sum_{k \in in(v)} |W(k,v)| \cdot |(f^\infty(k) - f^t(k))|}{|in(v)|}\Big) \leq$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(\frac{1}{|in(v)|}\sum_{k \in in(v)} \frac{1}{2^t}\Big) \leq$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(\frac{1}{|in(v)|}\frac{|in(v)|}{2^t}\Big) =$$
$$\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{1}{2^t} = \frac{1}{2^{t+1}} \rightarrow |f^\infty(u) - f^{t+1}(u)| \leq \frac{1}{2^{t+1}}$$

Therefore, $|f^\infty(u) - f^t(u)| \leq \frac{1}{2^t}$. This means that for a fixed error bound, $\epsilon << 1$, after a certain number t' of iterations, the fairness scores $f^{t'}$ of vertices become close to f^∞ and the algorithm converges. Please see the proof for Proposition 2 for how the value of t' depends on ϵ [30].

Due to lack of space, the proof for the convergence of goodness scores is given in Appendix B [30]. ∎

Proof for Theorem 2: Uniqueness of Fairness and Goodness scores.

Proof: Let us first prove that fairness scores are unique. The proof for unique goodness scores follows.

Let the fairness scores not be unique. So, let u be the node with maximum fairness difference, D (with $D \geq 0$), between its two possible scores $f_1^\infty(u)$ and $f_2^\infty(u)$

$$D = |f_1^\infty(u) - f_2^\infty(u)| =$$
$$|\Big(\frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f_1^\infty(k)}{|in(v)|}|\Big) -$$
$$\Big(\frac{1}{2|out(u)|}\sum_{v \in out(u)} |W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f_2^\infty(k)}{|in(v)|}|\Big)|$$
$$= |\frac{1}{2|out(u)|}\sum_{v \in out(u)} \Big(\Big(|W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f_1^\infty(k)}{|in(v)|}|\Big) -$$
$$\Big(|W(u,v) - \frac{\sum_{k \in in(v)} W(k,v) f_2^\infty(k)}{|in(v)|}|\Big)\Big)|$$

As $|x| - |y| \leq |x - y|$, applying it to the two terms within the summation, we have

$$D \leq |\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} |W(k,v)(f_2^\infty(k) - f_1^\infty(k))|}{|in(v)|}|$$

At this point, as $|x \cdot y| \leq |x| \cdot |y|$, we can write

$$D \leq |\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} |W(k,v)| \cdot |f_2^\infty(k) - f_1^\infty(k)|}{|in(v)|}| \leq$$

Now since $|W(k,v)| \leq 1$ and $|f_2^\infty(k) - f_1^\infty(k)| \leq D$, as D is the maximum fairness difference, then

$$D \leq |\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{\sum_{k \in in(v)} D}{|in(v)|}| =$$
$$|\frac{1}{2|out(u)|}\sum_{v \in out(u)} \frac{D \cdot |in(u)|}{|in(v)|}| = |\frac{1}{2|out(u)|}\sum_{v \in out(u)} D| = \frac{D}{2}$$

Thus, by solving $D \leq D/2$ and with the condition that $D \geq 0$, we obtain $D = 0$. Then, $|f_1^\infty(u) - f_2^\infty(u)| = 0$ and the fairness score is unique for each node in the network.

Since fairness scores are unique, and goodness scores will be fixed for fixed fairness scores, therefore, goodness score for each vertex is also unique. ∎