CE7490 Project: Benchmarking Algorithms for Weight Prediction in Weighted Signed Networks

Ruihang Wang
G1901564E
School of Computer Science Engineering
Nanyang Technological University
`ruihang001@e.ntu.edu.sg`

Meng Shen
G1902579B
School of Computer Science Engineering
Nanyang Technological University
`meng005@e.ntu.edu.sg`

Yihang Li
G1901408C
School of Electrical and Electronic Engineering
Nanyang Technological University
`leah9704@gmail.com`

Abstract

A number of weighted signed networks (WSN) exist in our world while many of them are incomplete. We are interested in finding the relationship and even capturing the scores between network users without direct connection. In this project, we first review a list of literatures that are related to signed social network. To predict weights of edges in such networks, we gather a wide collection of existing edge weight prediction algorithms and test their performance on real-world datasets. Experiments of leaving one edge and leaving $N\%$ edeges are conducted, seperativly on each social network. The benchmarking results indicate that the combination of all the features of existing algorithms to learn a linear regression model outperforms the past techniques for edge weight prediction. To make it avaliable for interested researchers, we open-source our benchmark in `https://github.com/RuihangWang/CE7490-OSN-Project`

1 Introduction

With the rapid development of information technology, social network has become a significant part of modern life. As remarkable number of messages reflecting opinion and attitude flowing on the network, social network analysis gradually gains popularity in a wide range of fields including political science, social psychology and business analysis. Social network analysis offers a tool to seek for insight through the structure of social networks but is limited by the complexity. Users are developing rich relationships through networks while analysis tools are generally reducing the relationships to simple links. To cover the gap between simplicity of network simulation and complexity of real relationships is a remarkable research topic.

In social network, people show different attitudes towards connection to others. The richness of a social network contains positive and negative interactions. The main part of connections are positive ones, including connecting with friends, fans, and followers. There also exist negative connections, including scammers, political enemy and opponents. These connections can be used to predict relationships between independent users. For 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 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 [?], anomaly detection[?] [?], network analysis[?] [?], community detection [?], information diffusion [?] [?] and sentiment prediction[?]. Therefore, the prediction of edge weights in WSNs can be advantageous in various tasks, both when WSNs are explicit and implicit.

To better understand theories and master practical skills, we investigate and implement a common set of algorithms and evaluate their performance on real-world datasets. The rest of the paper is structured as follows: Section 2 presents related work in this topic. Section 3 describes the overview of our project. Section 4 formulates the problem of weight prediction in weighted signed networks. Section 5 conducted experiments on real-world datasets using different algorithms. Section 6 evaluates the performance of all tested methods. The conclusion is summarized in Section 7.

2 Related Work

2.1 Edge Sign Prediction in SSNs

The very early work in SSN mainly based on balance and status. The first of these theories is structural balance theory, which originated in social psychology in the mid-20th-century. As formulated by Heider in the 1940s[?], and subsequently cast in graph-theoretic language by Cartwright and Harary[?], structural balance considers the possible ways in which triangles on three individuals can be signed, and posits that triangles with three positive signs and those with one positive sign are more plausible and hence should be more prevalent in real networks than triangles with two positive signs or none. Balanced triangles with three positive edges exemplify the principle that the friend of my friend is my friend, whereas those with one positive and two negative edges capture the notions that the friend of my enemy is my enemy, the enemy of my friend is my enemy, and the enemy of my enemy is my friend. Structural balance theory has been developed extensively in the time since this initial work[?], including the formulation of a variant weak structural balance proposed by Davis in the 1960s [?]as a way of eliminating the assumption that the enemy of my enemy is my friend. Weak structural balance posits that only triangles with exactly two positive edges are implausible in real networks, and that all other kinds of triangles should be permissible.

Balance theory can be viewed as a model of likes and dislikes. However, as Guha et al. observe in the context of Epinions[?], a signed link from P1 to P2 can have more than one possible interpretation, depending on P1s intention in creating the link. In particular, a positive link from P1 may mean, P2 is my friend, but it also may mean, I think P2 has higher status than I do. Similarly, a negative link from P1 to P2 may mean P2 is my enemy or I think P2 has lower status than I do.

Later work have developed features and models to predict the sign of an edge random-walk and trust-propagation algorithms[?][?], and using social interaction information for prediction [?][?]. Metric Multidimensional Scaling (MDS) 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 fairness and goodness method predicts the weight of an edge along with its sign.

2.2 Edge Weight Prediction in Social Networks

There is substantial work on predicting edge weights in unsigned social networks. Communication based features have been shown to be very important in quantifying the edge weights between users[?][?] [?][?], 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[?] and triadic balance and status measures[?] [?]. Two popular unsupervised algorithms are EigenTrust[?] and TidalTrust[?]. 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 sinks predecessors till it reaches the source. A very recent work in this direction is trustingness and trustworthiness[?]. However, these papers deal with edge weight prediction in unsigned social networks, while WSN is a new problem deriving from SSN. So in this paper, these techniques are implemented and compared with fairness and goodness method as benchmark models.

3 Project Overview

In this project, we extensively investigate and experiment methods for edge weight prediction in weighted signed networks. All algorithms are tested and evaluated on published real-world datasets. Moreover, we try our best to improve the performance on some traditional methods. The finished works are summarized as follows:

1. Literature review on OSN and select a topic about predicting weight of edges for weighted signed network.
2. Investigate the state-of-the-art algorithms *fairness-goodness* in[**?**] and studied a common set of baselines for weight prediction.
3. Conducted experiments on each algorithm and reproduce the results mentioned in original paper using real-world dataset.
   - *Experimental 1 - Removing one edge prediction*: When we remove one of the edges from the network and try to predict it from the rest of network structure one at a time, we compared fairness and goodness metrics with other typical methods and noticed that fairness and goodness method produces almost the best result.
   - *Experimental 2 - Removing N %-out edge prediction*: When we remove 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.
4. Evaluate results of different methods

4 Algorithms

In this section, we investigate a common set of algorithms associated with signed social networks. The procedures and formulas of these algorithms for edge weight prediction are discussed in details.

4.1 Fairness and Goodness

The fairness and goodness models is the state-of-art algorithm for edge weight prediction created by Srijan Kumar in [**?**]. The metrics are based on the intuition that 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. From the description above, we can see that the fairness and goodness metrics are dependent on each other.

To investigate this method, we first need to understand the defination of fairness and goodness from a mathmatical perspective. According to the author, the fairness and goodness dependency should satisfy the following definations.

Goodness defination: The goodness axiom states that vertices with higher fairness have higher impact on the vertices they rate. Formally, let u_1 and u_2 denote two vertices having identical ego-in-networks, and h be the 1-to-1 mapping between the two ego-in-networks. 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 $g(u_1) \leq g(u_2)$

Fairness defination: The fairness axiom states that a vertex is more fair than another if it gives ratings closer to the rating deserved by the recipent. Formally, let u_1 and u_2 denote two vertices having identical ego-out-networks, and h be the 1-to-1 mapping between the two ego-out-networks. 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 $f(u_1) \geq f(u_2)$.

Metrics Equations: From the above defination, the goodness ($g(v)$) and fairness ($f(u)$) are iteratively calculated by:

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

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

where $R = 2$[1], $in(v)$ is a set of nodes that preceed v, $out(u)$ is a set of nodes that succeed u, and $W(u,v)$ is the weight of (u,v). We say that two nodes u and v have the identical ego-in-network if $|in(u)| = |in(v)|$, Similarly, the ego-out-network is defined by the same rule. Therefore, the algorithm for calculating fairness and goodness is implemented as below:

```
Algorithm 1: Fairness and Goodness algorithm
Input: A WSN $G = (V, E, W)$
Output: Fairness and Goodness scores for all vertices in $V$
1 Let $f^0(u) = 1$ and $g^0(u) = 1, \forall u \in V$
2 $t = -1$
3 repeat
4     $t = t + 1$
5     $g^{(t+1)}(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f^t(u) \times W(u,v), \forall v \in V$
6     $f^{(t+1)}(u) = 1 - \frac{1}{2|out(v)|} \sum_{v \in out(u)} W(u,v) - g^{(t+1)}(v), \forall u \in V$
7 until $\sum_{u \in V} |f^{(t+1)}(u) - f^t(u)| > \epsilon \quad or \quad \sum_{u \in V} |g^{(t+1)}(u) - g^t(u)| > \epsilon$;
8 return $g^{(t+1)}(u) \quad and \quad f^{(t+1)}(u)$
```

Figure 1 shows the fairness and goodness distribution for all the vertices in the **BTCAlphaNet**. We can see that most vertices in the network 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.

Based on the definations and equations given by the original paper, we reproduce the edge weight prediction by taking the product of the fairness and goodness of each node. Thus, the predicted weight of (u,v) is given by:

$$W(u,v) = f(u) \times g(v) \tag{3}$$

where $W(u,v)$ depends on both the fairness $f(u)$ of the edge generator (u) and the goodness of the edge recipent (v).

4.2 Reciprocal

As a baseline of weight prediction, reciprocal algorithm takes the weight of (v,u) as equal to the weight of (u,v). When the reciprocal edge doesn't exist, the weight of that edge is set to 0. Therefore, the weight of (u,v) is predicted by:

$$W(u,v) = \begin{cases} W(v,u), & \text{if } u \rightarrow v \text{ exist} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

4.3 Triadic balance

This definition is derived directly from balance theory according to []. The algorithm takes 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. To be specific, let U_n and V_n denotes the set of neighbors of vertex u and v, repectively. To find all possible triads, vertexes

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

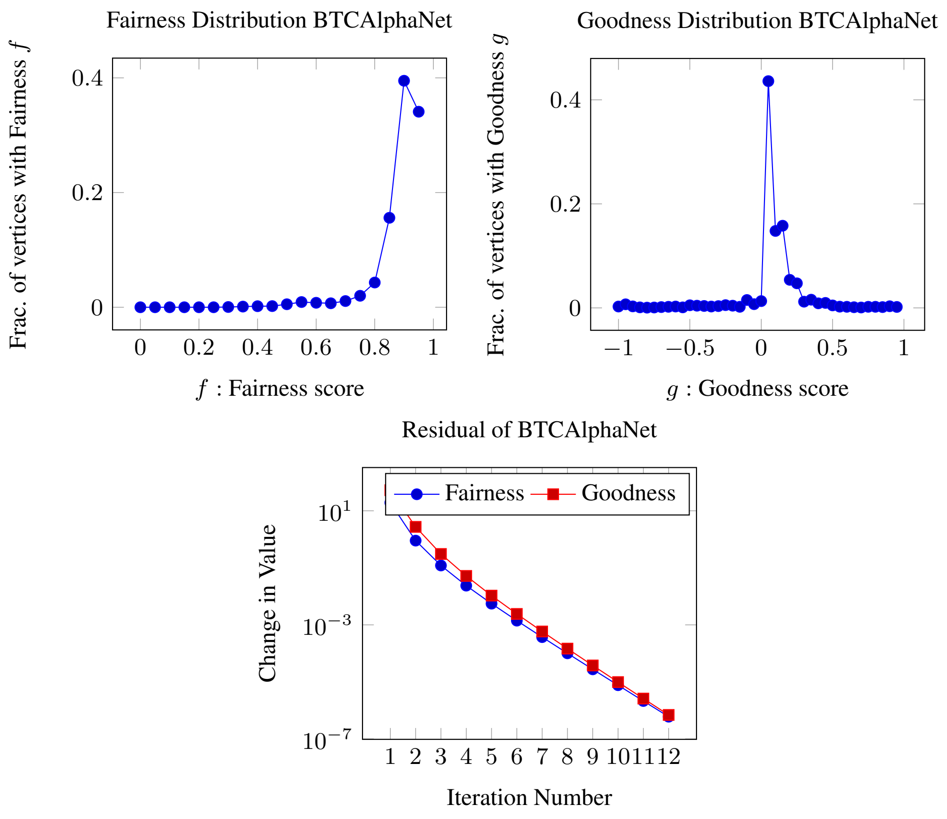


Figure 1: BTCAlphaNet

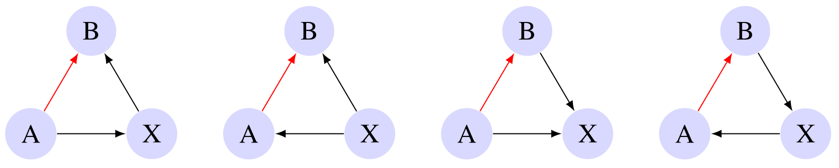
with both connections of u and v are obtained by $N = U_n \cap V_n$. When $N = \emptyset$, the weight of (u, v) is set to 0. Then, the weight of (u, v) is predicted by:

$$W(u,v) = \begin{cases} \frac{\sum_{n \in N} W(u,n)+W(n,u)+W(v,n)+W(n,v)}{M}, & \text{if } N \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where M is number of vertexes in set N, $W(u, n), W(n, u), W(v, n), W(n, v)$ are weights of each edge connected to u or v of vertexes in set N, repectively.

4.4 Triadic Status

Imaging there are two nodes named A and B, and they are unlinked. What we want to do is to predict the weight of the edge from A to B. Suppose that there is another node X, and (A, B, X) compose a triangle once A and B are linked. Overall, there are four different triangle-links, as the edge between X and A can go in either direction and similarly for the edge between X and B, if the sign of weight is not considered.



(a) t1 (b) t2 (c) t3 (d) t4

Figure 2: Triadic Status

And we predict the weight of the edge (A, B) using three rules:

1. In Graph t1, the predicted weight of the transitive edge is the sum of the two non-transitive edge weights.
2. In Graph t2 and t3, the predicted weight of a non-transitive edge is the difference of the transitive and the other non-transitive edge weights.
3. In Graph t4, since it is a cyclic triad, the weight of the missing edge is the negative of the sum of other two edge weights.

In the end, we take the average of predicted weights over all triadic-links that these two nodes are a part of.

4.5 Status Theory

Status theory is derived from []. The prediction made by this measure is the difference between the status of vertex u and vertex v. The status $\sigma(i)$ of a vertex i is defined as $\sigma(i) = |W^+_{in}(i)| - |W^-_{in}(i)| + |W^-_{out}(i)| - |W^+_{out}(i)|$. Status 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 us status is compared to vs.

4.6 Bias and Deserve

This method is proposed by Mishra and Bhattacharya in [**?**]. To compute "bias" and "deserve", we should first normalize the ratings (weights), and keep them in the range of [-1, 1] where 0 is a neutral opinion. Then, we say node u gives a trust-score of w_ij to node v for a given rating of (u, v). The two attributes of a node are defined by:

- *Bias*: This reflects the expected weight of an outgoing edge.
- *Deserve*: This reflects the expected weight of an incoming edge from an unbiased vertex.

Let $d^o(u)$ denotes the set of all outgoing edges from vertex u and likewise, $d^i(u)$ denotes the set of all incoming links to node u. Then, bias (BIAS) and deserve (DES) are iteratively computed as:

$$BIAS^{(t+1)}(u) = \frac{1}{2|d^o(u)|} \sum_{v \in d^o(u)} [W(u,v) - DES^t(v)] \tag{6}$$

$$DES^{(t+1)}(u) = \frac{1}{2|d^i(u)|} \sum_{v \in d^i(u)} [W(v,u)(1 - X^t(v,u))] \tag{7}$$

where $X^t(v,u) = max\{0, BIAS^t(v) \times W(v,u)\}$. The interative formulations of bias and deserve allow us to predict the weight of (u,v) based on the deserve value $DES(v)$ of vertex v. Thus, the weight is directly predicted by:

$$W(u,v) = DES(v) \tag{8}$$

4.7 Page Rank

We first compute Page Rank (ωPR) value of every vertex(k) in an unsigned graph according to [**?**]:

$$\omega PR(k) = \frac{1-\alpha}{|V|} + \alpha \sum_{z \in in(k)} \frac{\omega PR(z) * W(z,k)}{|out(z)|} \tag{9}$$

α is the probability called damping factor meaning a person will continue clicking on links(travelling in graph), and we set α to 0.85. $|V|$ is the number of all nodes. Eventually, $\omega PR(k)$ will converge to the probability that a person will go and stay at node k.

Then we compute edge weight of (u, v) using weighted average of PageRank values:

$$W(u,v) = \frac{\sum_{z \in out(u)} \omega PR(z) * W(u,z) + \sum_{z \in in(v)} \omega PR(z) * W(z,v)}{\sum_{z \in out(u)} \omega PR(z) + \sum_{z \in in(v)} \omega PR(z)} \tag{10}$$

4.8 Signed-Hits

The prediction is computed by using a modified version of HITS for signed network, called Signed-HITS[**?**]. Signed-HITS will compute the hub and authority scores of every node separately on positive graph(all edges' weights are positive) and negative graph, using the euqation:

$$\begin{aligned} h^+(u) &= \sum_{v \in out^+(v)} a^+(v); a^+(u) = \sum_{v \in^+(u)} h^+(v) \\ h^-(u) &= \sum_{v \in out^-(v)} a^-(v); a^-(u) = \sum_{v \in^-(u)} h^-(v) \end{aligned} \tag{11}$$

after convergence, we assign the authority score $a(u) = a^+(u) - a^-(u)$ and hub score $h(u) = h^+(u) - h^-(u)$ to each vertex u. Authority scores estimate the node value based on the incoming links, while hub scores estimate the node value based on outgoing links. However, the paper[**?**] didn't deliver the method to compute edge weight by using the authority score and hub score. Again, we use weighted average to compute the weight prediction:

$$W(u,v) = \frac{h(u) * \sum_{z \in out(u)} W(u,z) + a(v) * \sum_{z \in in(v)} W(z,v)}{h(u) * |V_{z \in out(u)}| + a(v) * |V_{z \in in(v)}|} \tag{12}$$

In this equation, we use $h(u)$ to be the weight of all out-going edges and $a(v)$ to be the weight of all in-coming edges. The reason is that $h(u)$ represents the node value based on out-going edges. So the predicted weight of out-edges from u should be higher if $h(u)$ is high. Likewise, the predicted weight of in-edges into v should be higher if $a(v)$ is high.

4.9 Linear Regression

In this experiment, we use a linear regression model to predict the weight of edge. Our features are generated from some algorithms mentioned above. First, the test edge e_t whose weight is to be predicted is removed from the network. Second, we remove the second edge e_s and predict the weight of e_s from the rest of the edges. And the predicted value of e_s will be the feature and the true value of e_s will be the ground truth. The second step is done multiple times to generate enough samples for training. After training, we use this model to predict the value of the removed edge e_t.

5 Performance Evaluation

We implemented two series of experiments to evaluate the performance of fairness and goodness method on three public WSN dataset mentioned above.

Experiment1:leave-one-out prediction In these experiments, one weighted edge is removed at a time and nine algorithms are implemented to predict the weight of test edge, simulating the situation where we want to calculate the weight of a non-existed edge. The root-mean-square error (RMSE) and Pearson correlation coefficient (PCC) are calculated in each loot and the average result is shown in table **??**. Fairness and goodness method achieved almost the best performance among all datasets and showed robustness. The linear regression result trained by all other features is the best one.

Experiment2:leave-N%-out prediction In this part, we removed a larger number of edges to further evaluate the performance of F&G algorithm. We randomly selected N% of the edges, remove them as test edges and predict their weight from the rest of network. The value of N range from 10 to 90 with a step of 10. Figure**??**, Figure**??** and Figure**??** showed average RMSE and PCC results in the three datasets. F&G method beats most of the algorithms and linear regression method showed robustness and performed nearly the best in all datasets. We noticed that performance of linear regression method showed stability when the percentage of removed edges varies, meaning that this method can handle networks partly invisible, especilly online networks constrained by privacy items.

Table 1: Performance of different algorithms for edge weight prediction in leave-one-out experiment on different datasets. Each cell is (RMSE, PCC). Lower RMSE and higher PCC are better.

Algorithms	**BTCAlphaNet**	**OTCNet**	**RFANet**
	Existing Algorithms		
PageRank	(0.29,0.25)	(0.35,0.30)	(0.21,0.48)
Bias Deserve	(0.23,0.60)	(0.28,0.61)	(0.21,0.48)
Reciprocal	(0.28,0.44)	(0.33,0.47)	(0.35,0.00)
Signed HITS	(0.23,0.66)	(0.27,0.69)	(0.23,0.44)
Status Theory	(0.45,0.00)	(0.48,-0.01)	(0.39,-0.15)
Triadic Balance	(0.31,0.21)	(0.36,0.28)	(0.24,0.29)
Triadic Status	(0.35,0.24)	(0.37,0.42)	(0.32,0.27)
	Fairness and Goodness		
Fairness and Goodness	(0.24,0.60)	(0.28,0.63)	(0.22,0.46)
	Linear Regression		
Linear Regression	(0.22,0.67)	(0.25,0.72)	(0.20,0.54)

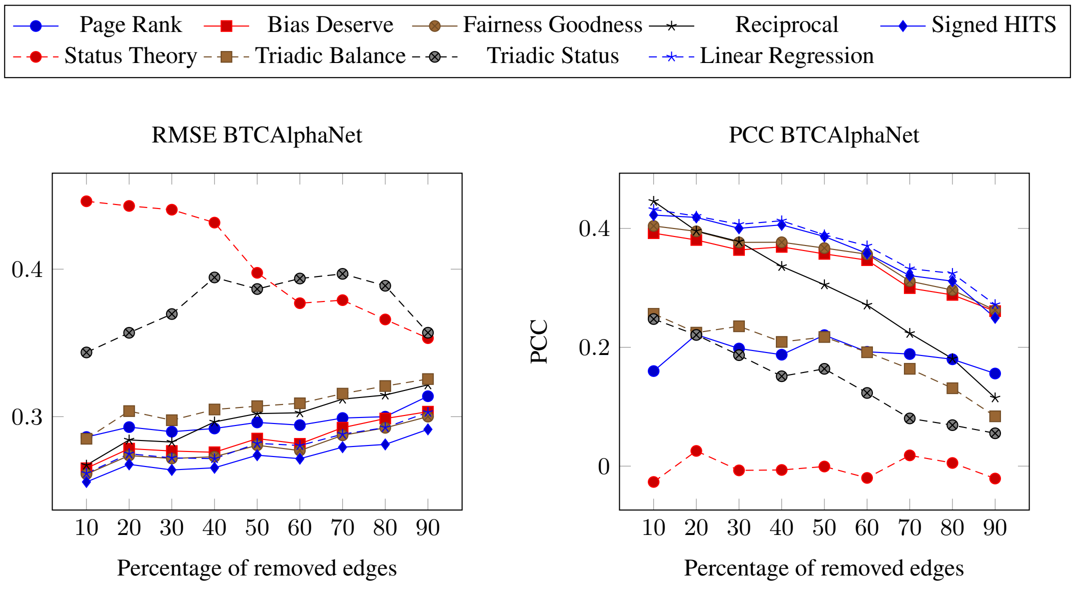


Figure 3: BTCAlphaNet

6 Conclusion

Appendix

A. Real-World Datasets

We thank the authors of [] who release the datasets online for research. The real-world datasets used for evaluation are summarized in Table 2. The listed networks are all 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 normalized in the range of $[-1, 1]$. The meaning of each network is illustrated as follows:

Bitcoin networks. As a cryptocurrency ussed for anonymous trading, Bitcoin is always accompanied with certain risk. Therefore, users often rate the level of trust they have in others. Based on such

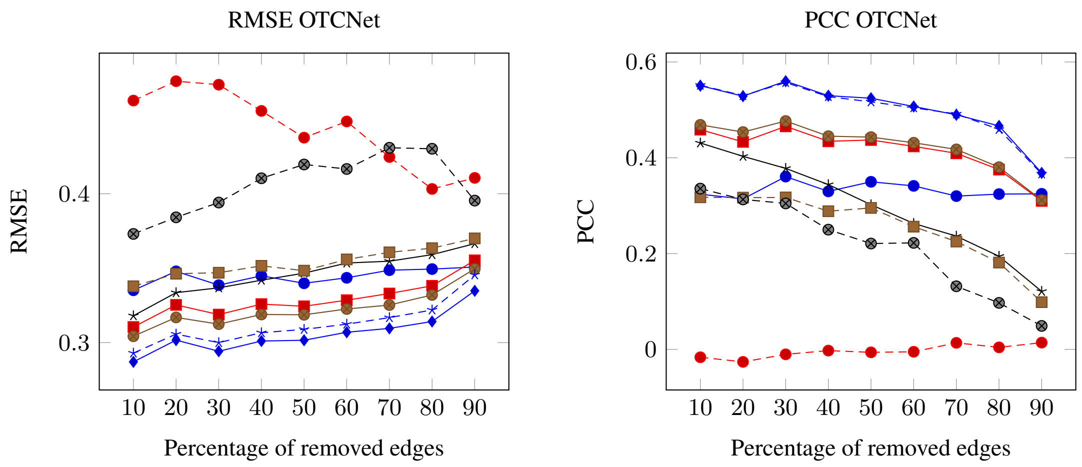


Figure 4: OTCNet

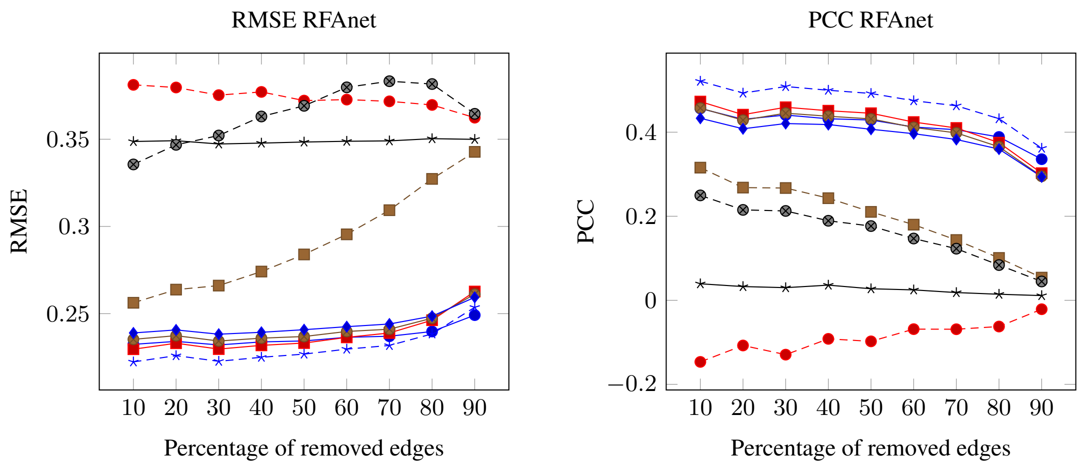


Figure 5: RFAnet

exchange, Bitcoin-OTC and Bitcoin-Alpha[2] are gatherd on a scale between -10 and 10. -10 means the person at that node may like be a fraudster, while 10 means a person can be most trusted. Thus, these two exchanges explicitly yield WSNs.

Wikipedia RfA Wikipedia Requests for Adminship (RfA) network 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. To normalize this datasets in the range of $[-1, 1]$. An online sentiment engine is used to quantify the strength of each score based on the intensity of the sentiment expressed in its explanation.

Table 2: Real-world datasets used for evaluation

Network	**Vertices**	**Edges**	**Pos. Edges** %	**Description of network edges**
Bitcoin-Alpha	3,783	24,186	93%	Degree of trust or distrust from Bitcoin user u to v
Bitcoin-OTC	5,881	35,592	89%	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 electio

A. Fairness and Goodness Distribution

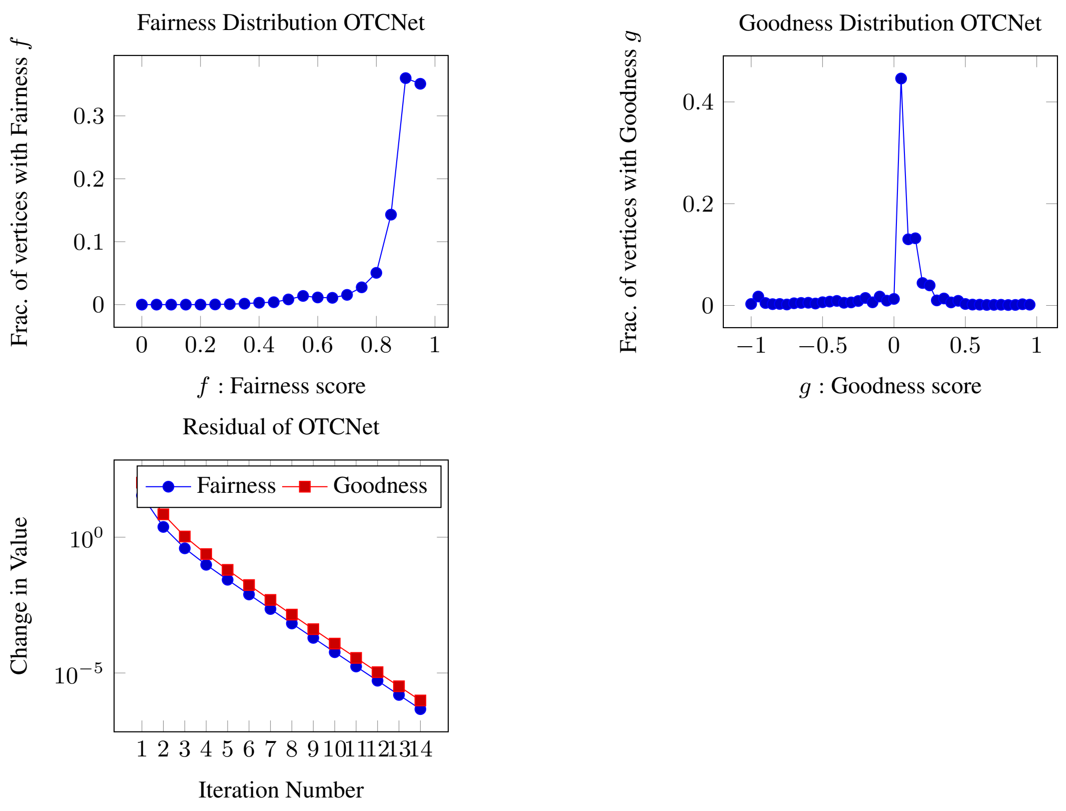


Figure 6: OTCNet

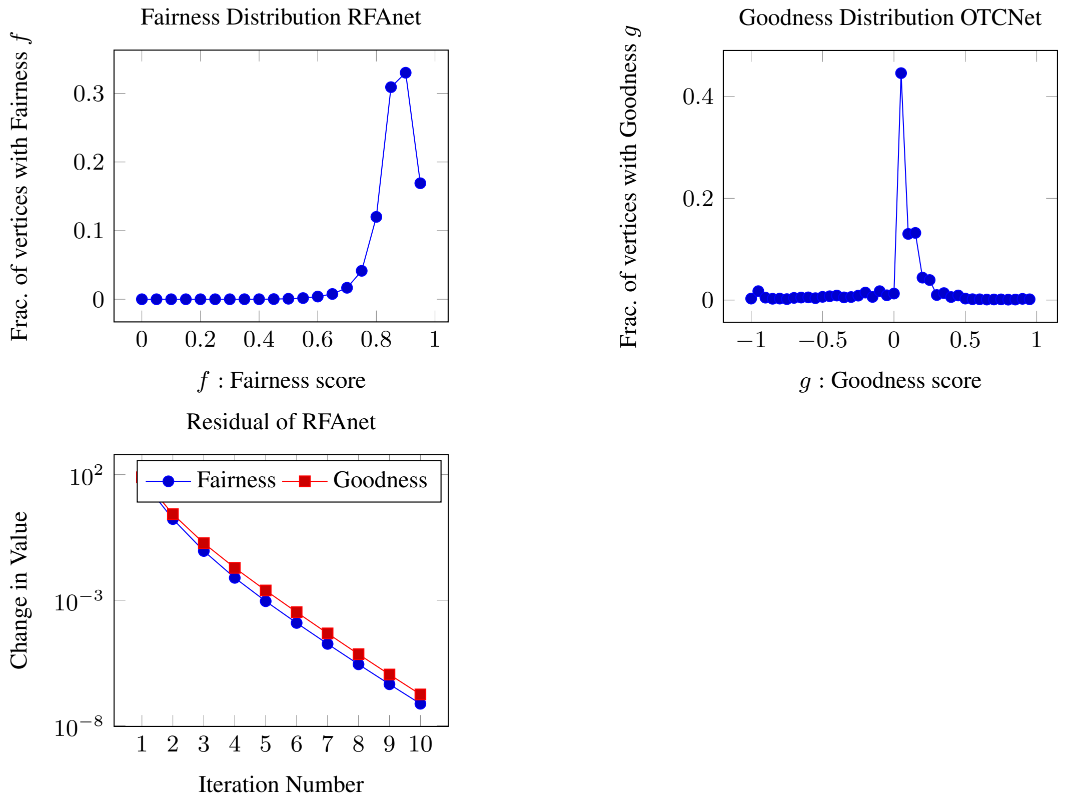


Figure 7: RFAnet