On the Optimality of Trust Network Analysis with Subjective Logic

Yongsu PARK
Division of Computer Science and Engineering, Hanyang University, Korea
yongsu@hanyang.ac.kr

***Abstract*—Building and measuring trust is one of crucial aspects in e-commerce, social networking and computer security. Trust networks are widely used to formalize trust relationships and to conduct formal reasoning of trust values. Diverse trust network analysis methods have been developed so far and one of the most widely used schemes is TNA-SL (Trust Network Analysis with Subjective Logic). Recent papers claimed that TNA-SL always finds the optimal solution by producing the least uncertainty. In this paper, we present some counter-examples, which imply that TNA-SL is not an optimal algorithm. Furthermore, we present a probabilistic algorithm in edge splitting to minimize uncertainty.**

***Index Terms*—Trust, Reputation, Subjective logic, Trust networks, Identity management system**

I. Introduction

Nowadays smartphones are widely being used in our society and traditional off-line social activities are being merged into the on-line computing environments, e.g., social networking or e-commerce.

Meanwhile, continuous growth of complex on-line networking activities can arouse security concerns. Trust is a key aspect of relationship between entities and measuring trust is one of the most important prerequisites for establishing the safe and reliable e-society. However, it is considered that measuring trust is still yet complex and little understood [2].

A trust metric is a measurement of the degree to which one entity trusts another. For measuring trust, empirical metrics use surveys or game-like scenarios to capture values of trust. Formal metrics focus on formal trust representation and modelings, and furthermore formal reasoning about trust, by using algebra, probability or logic.

A trust network is used to formalize trust relationships and to conduct formal reasoning of trust values [2-4]. Trust networks consist of nodes to represent peers and directed links to represent trust-relationships.

Up to now, a considerable number of trust management systems and reputation (that is closely related to trust but not exactly the same) management systems have been proposed [5]: PageRank [6], Eigentrust [7], Mocatrust, KeyNote, TBAC, Subjective Logic, [8], [9], etc. Among them this paper focuses on Subjective Logic [10-11] and TNA-SL [3] trust network, which is one of widely and actively used for analyzing social networking services, e-commerce, approximate reasoning, etc.

The main purpose of TNA-SL is modeling transitive trust relationship between entities using subjective logic and finding/calculating the most confident combined trust values [3]. [1] claimed that experimental results of edge splitting and TNA-SL coincide with each other and that TNA-SL always finds the optimal solution by producing the most confident value (a.k.a. the lowest uncertainty value). However, in this paper we present some counter-examples, which imply that TNA-SL is not an optimal algorithm. Furthermore, we present an efficient probabilistic algorithm to calculate the near-optimal parameters in edge splitting.

This paper is organized as follows. Section 2 describes related work, overview of trust networks, and structured notation. Section 3 briefly explains subjective logic and some operators used in subjective logic. Section 4 describes network simplification methods in trust networks with subjective logic. Section 5 shows some counter-examples for the TNA-SL algorithm and then presents the proposed scheme in edge-splitting, which is based on Nelder-Mead method. Section 6 concludes the paper.

II. Overview of Trust Networks

As mentioned in Section 1, recently a large amount of distributed trust/reputation management schemes have been published in which each peer organizes an initial trust group and gradually extends the group and updates trust information [6-7],[12-15] to provide personalized trust information in distributed environments. However, such schemes seem infeasible since the underlying relation graph cannot be used for general cases or they have an exponential order of time complexity [16]. We omit description of each work due to lack of space (please refer to [5] for summary of some major schemes). This paper focuses on Subjective Logic [10-11] and TNA-SL [3] trust network, which is one of widely and actively used.

Before describing TNA-SL, Section 2.1 describes trust graph with subjective logic and Section 2.2 explains structured notation.

A. Trust graph with subjective logic

Recall that trust networks consist of nodes to represent peers and links to represent trust-relationships [17]. TNA-SL (Trust Network Analysis using Subjective Logic) uses subjective logic to represent trust relation. In TNA-SL, there are two types of trust relation: *transitive trust* and *parallel trust combination*.

Transitive trust is the case where someone, Alice, trusts another person, Carol, by the judgments of the already established trust relationships, i.e., from Alice to Bob and from Bob to Carol. Fig. 1 shows an example of transitive trust. In this case, trust relation between Alice and Bob is

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012R1A1A2007263).

Digital Object Identifier 10.4316/AECE.2014.03006

different from that between Bob and Carol where the former is closely related to recommendation. We call the former referral trust and the latter functional trust. Transitive trust can be considered as an indirect trust where the degree of indirection can be extended multiple times.

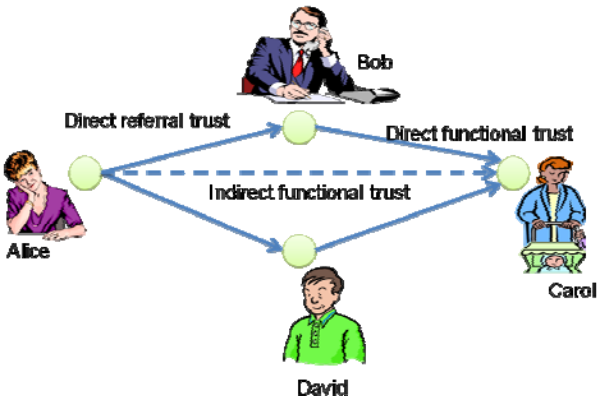

Figure 1. An example of transitive and parallel trust combination.

In the trust network, if there is a path from a node A to another B, we can think that there is a transitive trust from A to B. However, generally transitive trust is not always applied because trust scope is different, e.g., if A trusts B for taking care of a baby and B trusts C for fixing a car, trust is not transitive. Transitive trust is considered to be important to improve quality of trust/recommendation because even for millions of users direct trust is made only among limited users [17].

In the trust network, trust can be combined from information of different sources to make better decisions. This can be modeled as *parallel trust combination*. Suppose that A trusts both B and D and that B recommends C to A (for something), A may would like to get the second opinion for C, e.g., get another D's recommendation for C. If, both recommendations are positive, A can trust more on C. Otherwise, if A receives conflicting recommendations, A may trust less on C. We will use subjective logic to combine trust values, which handles for all these cases.

B. Structured notation

Transitive trust networks can be expressed some notations called *structured notation* [3]. First, the direct edge from A to B is expressed as $[A,B]$. The symbol ":" is used to denote the transitivity, e.g., if A trusts B and B trusts C, $([A,C]) = ([A,B]:[B,C])$. For combining trust, we use "◊"symbol, e.g., in the previous example, if there is an additional path from A to D and from D to C, $([A,C]) = ([A,B]:[B,C]) \lozenge ([A,D]:[D,C])$.

III. Subjective Logic

Subjective logic is a belief reasoning calculus which is an extension of probabilistic logic to modelize uncertainty, incomplete knowledge and different views. In subjective logic, *opinions* express subjective beliefs. There are two types of opinions: binomial opinion for a single proposition and multinomial opinion for multiple propositions. In this paper we focus on only binary opinions for lack of space.

Suppose that x be a proposition. A binomial opinion w_x on the truth of x is (b, d, u, a) where b (belief) is the belief that x is true, d (disbelief) is the belief that x is false, u (uncertainty) is the amount of uncommitted belief, and a (base rate) is the *a priori* probability in the absence of evidence. These components satisfy $0 \le b,d,u,a \le 1$ and $b+d+u=1$. An opinion with $b=1$ is equivalent to binary logic TRUE and that with $d=1$ is equivalent to FALSE. If $b+d=1$, the opinion is equivalent to the traditional probability and if $b+d<1$ it expresses degree of uncertainty.

The trust opinion of subjective logic is compatible with the reputation representation of Bayesian reputation systems, which are being widely used [3]. By using this fact, we can use Bayesian reputation systems to determine trust opinions, which is described in detail in [1],[3].

A. Transitivity [3],[18]

Suppose that $w^A_B=(b^A_B, d^A_B, u^A_B, a^A_B)$ and $w^B_C=(b^B_C, d^B_C, u^B_C, a^B_C)$ are the opinion from peer A to B and that from B to C, respectively. Then, the transitive opinion from A to C, $w^{A:B}_C$, which is denoted as $w^A_B \otimes w^B_C$, is calculated as follows: $w^{A:B}_C =(b^{A:B}_C, d^{A:B}_C, u^{A:B}_C, a^{A:B}_C)$ where $b^{A:B}_C = b^A_B b^B_C$, $d^{A:B}_C = b^A_B d^B_C$, $u^{A:B}_C = d^A_B + u^A_B + b^A_B u^B_C$, $a^{A:B}_C = a^B_C$.

B. Cumulative fusion [3]

Assume that there are two peers A and B who have observed C and their opinions are denoted by w^A_C and w^B_C, respectively. Then, cumulative opinion $w^{A\lozenge B}_C = w^A_C \oplus w^B_C$ is calculated as follows: $b^{A\lozenge B}_C = (b^A_C u^B_C + b^B_C u^A_C)/(u^A_C + u^B_C - u^A_C u^B_C)$, $d^{A\lozenge B}_C = (d^A_C u^B_C + d^B_C u^A_C)/(u^A_C + u^B_C - u^A_C u^B_C)$, $u^{A\lozenge B}_C = (u^A_C u^B_C)/(u^A_C + u^B_C - u^A_C u^B_C)$, $a^{A\lozenge B}_C = a^A_C$, where it is assumed that $a^A_C = a^B_C$. For $u^A_C = u^B_C = 0$, limits can be computed (for more details, refer to [1],[3]).

IV. Network Simplification in TNA-SL

Recall that structured notation contains 2 operators: ":, ◊." Since subjective logic can support both, if the transitive network is expressed by structured notation and all direct trust opinions are predefined, we can calculate the *combined trust opinion* using transitivity and cumulative function.

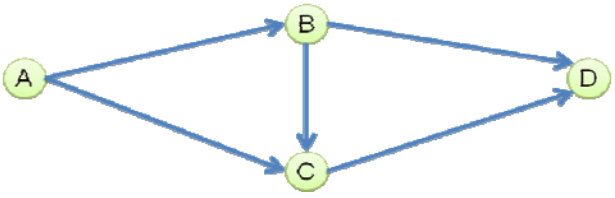

Figure 2. An example of the trust graph.

However, there is a problem to represent trust networks using structured notation, which is illustrated in Fig. 2. If we convert the network in this figure into structured notation, $([A,D])=(([A,B]:[B,D]) \lozenge ([A,C]:[C,D]) \lozenge ([A,B]:[B,C]:[C,D]))$ because it contains 3 paths from A to D.

The problem in this equation is that edges $[A,B]$ and $[C,D]$ appear twice. If we use this equation to compute the combined trust opinion $[A,D]$, trust opinions of $[A,B]$ and $[C,D]$ are used twice while the others are used only once. Some trust models allow multiple usages of the same edge (i.e. computed results are the same regardless of multiple usages) but unfortunately subjective logic does not allow that.

Hence, to use subjective logic, structured notation should be *canonicalized* [1],[3]. An expression of a trust graph in

structured notation where every edge only appears once is called *canonical*.

Unfortunately, a large amount of cases including Fig. 2 cannot be canonicalized. Hence, diverse graph simplification/modification techniques can be used for canonicalization.

[3] presents two algorithms for canonicalization. Both use the property that if the trust network is DSPG (Directed Series-Parallel Graph) [19], it is canonical [3]. The first one is called the optimal DSPG. The optimal DSPG focuses on computing the highest confidence level of the combined trust opinion, not computing the most positive/negative trust value. The most confidence level is achieved by finding the combined trust opinion such that it has the least uncertainty u [3].

First, it computes the set containing all the paths from the source node to the destination node. Then, for every subset it checks whether the subset can be represented as DSPG. If so, the combined trust opinion is computed using subjective logic. (Otherwise, that subset is ruled out.) Finally, it produces the subset containing the highest confidence (i.e., the lowest uncertainty of the trust opinion). This algorithm is called optimal because it always produces the subset containing the highest confidence. However, since the number of subsets is exponentially huge, the computation cost is too high.

The second algorithm is a near-optimal algorithm, called near-optimal DSPG. It is a heuristic-based algorithm where detailed description is out of scope of this paper (please refer to [3]). For clarity, in this paper we call the first algorithm as the algorithm of TNA-SL hereafter.

A. Edge splitting [1]

Recently, [1] presents the new method that is called *edge splitting* where for every pair of paths from the source node to the destination node, if they have the common edge(s), we insert virtual node(s) and split edge(s), i.e., one for the original node and the other for the virtual node. An example is shown in Fig. 3.

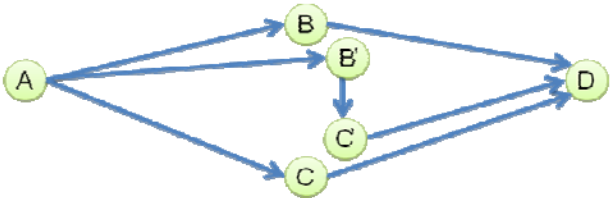

Figure 3. The results of edge splitting of Fig.2.

In edge splitting, the trust opinion w for each common original edge should be split into 2 opinions: w_1 and w_2. Equations for calculating w_1 and w_2 from w are as follows: $w_i = (b_i = (\phi_i b)/(\phi_i(b+d)+u))$, $d_i = (\phi_i d)/(\phi_i(b+d)+u))$, $u_i = u/(\phi_i(b+d)+u)$, $a_i = a$), where $0 \le \phi_1, \phi_2, \le 1$ and $\phi_2 = 1-\phi_1$. For simplicity, we call ϕ_1 as ϕ hereafter.

It can be verified that $w_1 \oplus w_2 = w$, as expected. ϕ is called the *fission factor* that determines the proportion of evidence assigned to each independent opinion part.

V. On the Optimality of TNA-SL

Section 5.1 describes some counter-examples for the optimality of TNA-SL and Section 5.2 presents the proposed edge-splitting scheme to minimize fission factors.

A. Counter-examples for the optimality of TNA-SL

[1] claims that when edge splitting is used in the trust network analysis, if we compute ϕ value such that the combined opinion has the lowest uncertainty value (i.e., the highest confidence), ϕ is always either 0 or 1. In this case, the path containing the corresponding edge (w_1 for ϕ=0, w_2 for ϕ=1) is ruled out because its combined opinion has the least confidence (i.e. the maximum uncertainty). Hence, the result is identical to that of the optimal DSPG algorithm [3] in TNA-SL. Hence, [1] claims that the optimal DSPG algorithm in TNA-SL is in fact the optimal algorithm.

However, we found some counter-examples that ϕ is neither 0 or 1. Moreover, we found the cases such that the uncertainty is strictly lower than the optimal DSPG, which implies that TNA-SL is not an optimal algorithm.

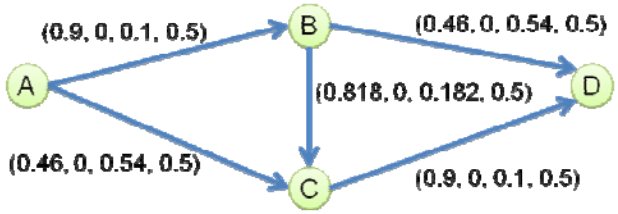

Figure 4. An example of the trust graph.

Fig. 4 shows such counter-examples. We set the trust opinion for each edge as in Fig. 4. In this figure, totally there are 3 paths: ([A,B]:[B,D]), ([A,C]:[C,D]), ([A,B]:[B,C]:[C,D]). Since there are two common edges: [A,B], [C,D], we split these and get the result as: ([A,B]:[B,D]), ([A,C]:[C,D]), ([A,B']:[B',C']:[C',D]).

We tested for various $\phi^A{}_B$ and $\phi^C{}_D$ values and calculated u of the combined opinion of (A,D), some of experimental results are shown in Fig. 5 (we show only $\phi=\phi^C{}_D=\phi^A{}_B$ cases for better understanding). In this example, the optimal ϕ is 0.25 and the uncertainty value is 0.28 where the trust opinion of ([A,D]) is (0.72, 0, 0.28, 0.5). The sub-optimal DSPG and optimal DSPG produce the results having only 2 paths, i.e., $\phi^C{}_D=\phi^A{}_B=1$ and in this case the uncertainty value is over 0.4.

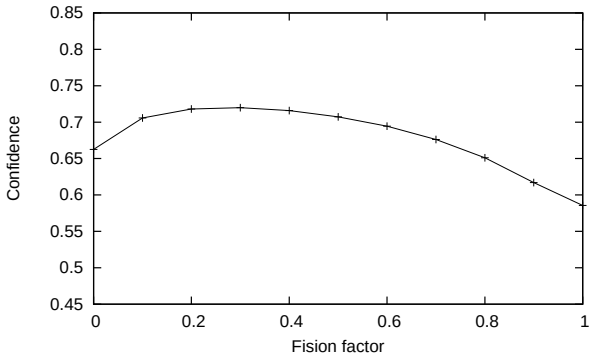

Figure 5. The case where ϕ is neither 0 nor 1.

To examine carefully, we conducted the experiments, where experimental environments are as follows: CPU, RAM, Operating system, and Programming language are Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz, 8 GBytes, SMP Debian 3.14.12-1 (2014-07-11) x86_64 GNU/Linux,

and gcc version 4.9.0. We used the network described in Fig. 4, where there are 5 edges. For subjective logic opinion w values for these edges, we assume that the value a in $w=(b,d,u,a)$ is the same for all edges to apply transitivisity/fusion operations. To analyze under which conditions TNA-SL is optimal, we consider the following different cases:

- **CASE 1:** for each edge, (b,d,u,a) are randomly generated from uniform distribution [0,1], where restriction is $b+d+u=1$.
- **CASE 2:** for each edge, (b,d,u,a) are randomly generated from uniform distribution [0,1], where restriction is $b+d+u=1$ and $u=0$.
- **CASE 3:** for each edge, (b,d,u,a) are randomly generated from uniform distribution [0,1], where restriction is $b+d+u=1$ and $d=0$.
- **CASE 4:** for each edge, (b,d,u,a) are randomly generated from uniform distribution [0,1], where restriction is $b+d+u=1$ and $b=0$.
- **CASE 5:** for all edges, we set the $b=1$ for opinions (which implies *d, u* are 0).
- **CASE 6:** for all edges, we set the $d=1$ for opinions (which implies *b, u* are 0).
- **CASE 7:** for all edges, we set the $u=1$ for opinions (which implies *b, d* are 0).

For each trial, we changed ϕ^{C}_{D} and ϕ^{A}_{B} values, from 0.00 to 1.00 at the step of 0.01, respectively, to find the minimal u value of the combined opinion of (A,D). If the lowest uncertainty u value is computed when ϕ^{C}_{D} and ϕ^{A}_{B} are either 0 or 1, respectively, for this case TNA-SL is optimal. Otherwise, we regard that for this case TNA-SL is not optimal. For each CASE (CASE 1~7), we repeated experiments for 1,000,000 times. Table I summarizes the experimental results.

TABLE I. EXPERIMENTAL RESULTS.

CASE	*Description*	*The probability that TNA-SL is NOT optimal.*	*The probability that TNA-SL is optimal.*
1	For all edges, w is randomly generated from uniform distribution [0,1], where restriction is $b+d+u=1$.	22.61%	77.39%
2	The same condition as CASE 1 except for an additional restriction: $u=0$.	57.10%	42.90%
3	The same condition as CASE 1 except for an additional restriction: $d=0$.	16.52%	83.48%
4	The same condition as CASE 1 except for an additional restriction: $b=0$.	92.33%	7.67%
5	For all edges, we set the $b=1$ for opinions (which implies *d, u* are 0).	0%	100%
6	for all edges, we set the $d=1$ for opinions (which implies *b, u* are 0).	0%	100%
7	for all edges, we set the $u=1$ for opinions (which implies *b, d* are 0).	0%	100%

In CASE 5, 6, and 7, we could not found any count-examples, which may imply that TNA-SL is optimal for these cases.

For CASE 1, for randomly generated values w for edges, the lowest uncertainty u is obtained where ϕ^{C}_{D} and ϕ^{A}_{B} are not 0 or 1 with the probability of 22.61%.

For CASE 3, for randomly generated values w for edges, the lowest uncertainty u is obtained where ϕ^{C}_{D} and ϕ^{A}_{B} are not 0 or 1 with the probability of 16.52%, which is differ from CASE 1. For CASE 2 and 4, the probability values are 57.10% and 92.33%, respectively. This may be due to different distribution of opinion values (w); we could not find the correct cause why they produce the different results.

Also, we repeated the same experiments for different a values and get the same results for all CASEs, which implies that the value a does not affect the experimental results for all CASEs.

Additionally, we provide some counter-example cases (from CASE 1) where the optimal ϕ^{C}_{D} (or ϕ^{A}_{B}) are not 0 and 1, which are shown in Table II. (we assume that $a=0.5$ in opinion values for all edges.)

TABLE II. SOME CASES WHERE ϕ IS NEITHER 0 NOR 1 FOR FIG. 4.

Case 1	b^{A}_{B}	b^{C}_{D}	b^{B}_{D}	b^{A}_{C}	b^{B}_{C}	ϕ^{A}_{B}	ϕ^{C}_{D}
	0.66	0.92	0.71	0.53	0.41	0.86	0.77
	d^{A}_{B}	d^{C}_{D}	d^{B}_{D}	d^{A}_{C}	d^{B}_{C}		
	0.26	0.02	0.28	0.42	0.57		
Case 2	b^{A}_{B}	b^{C}_{D}	b^{B}_{D}	b^{A}_{C}	b^{B}_{C}	ϕ^{A}_{B}	ϕ^{C}_{D}
	0.96	0.63	0.23	0.37	0.26	0.92	0.62
	d^{A}_{B}	d^{C}_{D}	d^{B}_{D}	d^{A}_{C}	d^{B}_{C}		
	0.03	0.29	0.65	0.58	0.31		
Case 3	b^{A}_{B}	b^{C}_{D}	b^{B}_{D}	b^{A}_{C}	b^{B}_{C}	ϕ^{A}_{B}	ϕ^{C}_{D}
	0.18	058	0.04	0.05	0.22	0.76	0.58
	d^{A}_{B}	d^{C}_{D}	d^{B}_{D}	d^{A}_{C}	d^{B}_{C}		
	0.79	0.02	0.57	0.92	0.14		
Case 4	b^{A}_{B}	b^{C}_{D}	b^{B}_{D}	b^{A}_{C}	b^{B}_{C}	ϕ^{A}_{B}	ϕ^{C}_{D}
	0.20	0.36	0.45	0.63	0.61	0.64	0.93
	d^{A}_{B}	d^{C}_{D}	d^{B}_{D}	d^{A}_{C}	d^{B}_{C}		
	0.66	0.53	0.02	0.17	0.24		
Case 5	b^{A}_{B}	b^{C}_{D}	b^{B}_{D}	b^{A}_{C}	b^{B}_{C}	ϕ^{A}_{B}	ϕ^{C}_{D}
	0.28	0.55	0.34	0.09	0.43	0.68	0.60
	d^{A}_{B}	d^{C}_{D}	d^{B}_{D}	d^{A}_{C}	d^{B}_{C}		
	0.11	0.35	0.10	0.62	0.13		

B. Efficient algorithm to find the minimal ϕ value

First, we precisely define the problem to find the optimal ϕ values as follows: given the trust graph for subjective logic, the objective is to compute the combined opinion from the source node s to the target node t with minimal u. Assume that there are n paths from s to t. If some of these paths share common edge(s), we call each of which e_1, e_2, …, e_k and corresponding fission factors as ϕ_1, ϕ_2, …, $\phi_{k'}$ to use edge splitting (Note that $k \neq k'$ because for some cases one edge should be split into multiple m (≥ 3) edges. In these cases, we should apply edge splitting m-1 times sequentially). Then, the goal is to find the values ϕ_1, ϕ_2, …, $\phi_{k'}$ such that after edge splitting the uncertainty u of the combined opinion from s to t is minimal.

This problem is called multi-dimensional minimization, which is one of the widely known problems in optimization. Among many algorithms [20][21][22][23], we use Nelder-Mead method [20] since getting the derivative of the function u is complex if the trust network is large. Because getting the global minimum value is a very difficult problem, we use probabilistic approach to iteratively find

the local minimum with random seeds, which is described in detail as follows.

Algorithm 1: the proposed algorithm finds $x = (\phi_1, \phi_2, \ldots, \phi_{k'})$ $(1 \le i \le k'+1)$ with the minimal uncertainty u.

STEP 1: Randomly generate $x_i = (\phi_1, \phi_2, \ldots, \phi_{k'})$ $(1 \le i \le k'+1)$. Assume that $\{\mu_r, \mu_e, \mu_{oc}, \mu_{ic}\} = \{1, 2, 1/2, -1/2\}$. $x(\mu)$ is defined as $x(\mu) = (1+\mu)x' - \mu(x_{k'+1})$.

STEP 2: Sort x_i that $u(x_1) \le u(x_2) \le \ldots \le u(x_{k'+1})$. Set fcount = $k'+1$.

STEP 3: While $u(x_{k'+1}) - u(x_1) > \tau$,

STEP 3-a: Compute $x' = (\Sigma_{i=1}^{k'} x_i)/k'$. Compute $x(\mu_r)$ and $f_r = u(x(\mu_r))$. fcount++. If fcount=kmax then exit.

STEP 3-b (Reflect): If $u(x_1) \le f_r < u(x_{k'})$, replace $x_{k'+1}$ with $x(\mu_r)$ and goto **STEP 3-g.**

STEP 3-c (Expand): If $f_r < u(x_1)$ then compute $f_e = u(x(\mu_e))$. fcount++. $f_e < f_r$, replace $x_{k'+1}$ with $x(\mu_e)$; otherwise replace $x_{k'+1}$ with with $x(\mu_r)$. Goto **STEP 3-g**.

STEP 3-d (Outside Contraction): If $u(x_{k'}) \le f_r < u(x_{k'+1})$, compute $f_c = u(x(\mu_{oc}))$. fcount++. If $f_c \le f_r$, replace $x_{k'+1}$ with $x(\mu_{oc})$ and goto STEP 3-g; otherwise goto **STEP 3-f.**

STEP 3-e (Inside Contraction): If $f_r \ge x_{k'+1}$ compute $f_c = u(x(\mu_{ic}))$. fcount++. If $f_c < u(x_{k'+1})$, replace $x_{k'+1}$ with $x(\mu_{ic})$ and goto STEP 3-g; otherwise goto STEP 3-f.

STEP 3-f (Shrink): If fcount ≥ kmax - k', exit. For $2 \le i \le k'+1$: set $x_i = x_1 - (x_i - x_1)/2$; compute $u(x_i)$.

STEP 3-g: Sort the vertices such that $u(x_1) \le u(x_2) \le \ldots \le u(x_{k'+1})$.

STEP 4: Repeat **STEP 1~3** for predefined times to get the global minimum x.

For better understanding, an example for this algorithm is provided for Fig. 4, as follows.

Example 1. Assume that the trust graph is given as in Fig. 4. Suppose that the threshold τ=0.03 and kmax = 100. In STEP 1 of Algorithm 1, k'=2 and following values are randomly generated from uniform distribution between 0 and 1, e.g., $x_1 = (\phi_1=0.5, \phi_2=0.4)$, $x_2 = (\phi_1=0.2, \phi_2=0.7)$, $x_3 = (\phi_1=0.4, \phi_2=0.5)$. In STEP 2, $u(x_i)$, the overall uncertainty value for x_i, are computed: $u(x_1)=0.289$, $u(x_2)=0.314$, $u(x_3)=0.400$. Fortunately, they are already sorted in the increasing order. Set fcount = 3. In STEP 3, because $u(x_3) - u(x_1) = 0.400 - 0.289 > \tau = 0.03$, in STEP 3-a, we compute $x' = ((0.5+0.2+0.4)/3, (0.4+0.7+0.5)/3) = (0.367, 0.533)$. $f_r = u(x(\mu_r)) = u(x(1)) = u(2*x' - 1*(x_3)) = u((0.334, 0.366)) = 0.281$. fcount=4. Because $f_r < u(x_1)$, goto STEP 3-c (Expand): $f_e = u(x(\mu_e)) = u(x(2)) = u(3*x' - 2*(x_3)) = u((0.301, 0.599)) = 0.296$. fcount=5. Because $f_e > f_r$, we replace x_3 with $x(\mu_r)$. In STEP 3-g, sorted x_i are as follows: $x_1 = (\phi_1=0.334, \phi_2=0.366)$, $x_2 = (\phi_1=0.5, \phi_2=0.4)$, $x_3 = (\phi_1=0.2, \phi_2=0.7)$. In STEP 4, we repeat this procedure and go to STEP 3. Note that in the previous time of STEP 3 execution, $u(x_i)$ are as follows: $u(x_1)=0.289$, $u(x_2)=0.314$, $u(x_3)=0.400$. This time, $u(x_i)$ are as follows: $u(x_1)=0.281$, $u(x_2)=0.289$, $u(x_3)=0.314$, which are smaller than those of previous iteration. In the next iteration, $x' = (0.345, 0.489)$, $f_r = u((0.489, 0.277)) = 0.2871$. Because $u(x_1) <= f_r < u(x_3)$, we replace x_3 with $x(1)$ and goto STEP 3-g. Now $x_1 = (\phi_1=0.334, \phi_2=0.366)$, $x_2 = (\phi_1=0.489, \phi_2=0.277)$, $x_3 = (\phi_1=0.5, \phi_2=0.4)$, where $u(x_1)=0.281$, $u(x_2)=0.287$, $u(x_3)=0.289$. In this way, after each iteration, $u(x_i)$ are getting smaller. If $u(x_1) - u(x_3) < \tau$ which implies that we have approached the minimum $u()$ value very closely, or if fcount >= kmax, which means we have already tried enough iterations, this algorithm terminates. □

Algorithm 1 is a probabilistic algorithm since in STEP 1, we initialize x_i with the random values and then start the execution. If we are fortunate, this algorithm finds the global minimum uncertainty value (and corresponding $x = (\phi_1, \phi_2, \ldots, \phi_{k'})$). Otherwise, it finds the local minimum uncertainty value. Hence, we should repeat Algorithm 1 for enough times to get the global minimum value with high probability. (To the best knowledge, there is no algorithm to get the global minimal value surely in the derivative-free multi-dimensional minimization problem).

As for efficiency of Algorithm 1, it is based on Nelder-Mead method. On multi-variables optimization, there exist significant amount of algorithms, e.g., BFGS (Broyden–Fletcher–Goldfarb–Shanno) [21], Flecher and Reeves [22], L-BFGS-B [23], etc. BFGS [21], L-BFGS-B [23], or other quasi-Newton methods require derivative of the target function, which is difficult if the network size is large and the network is complex. Flecher and Reeves [22] is an algorithm for the numerical solution of particular systems of linear equations.

Derivative-free methods can be classified into three: directional-direct search methods, line-search methods based on simplex derivative, and trust-region methods [24]. The second group requires partial derivatives while in the third group we should design a model to approximate the target function in the trust-region. In the first group, Nelder-Mead is the most representative method [24]. Even though there are variants for heuristics in Nelder-Mead, performance (how fast to find the minimum value) may differ from the structure of networks and opinion values of edges, which is the future work of this research.

VI. Conclusion

Trust networks are widely used to formalize trust relationships and to conduct formal reasoning of trust values. Diverse trust network analysis methods have been developed so far and one of the most widely used schemes is TNA-SL (Trust Network Analysis with Subjective Logic).

Recently, [1] claimed that TNA-SL always finds the optimal solution by producing the least uncertainty. In this paper, we first present some counter-examples, which imply that TNA-SL is not an optimal algorithm. Then, we present a probabilistic algorithm in edge splitting, which is designed to produce near-optimal fission factors by using Nelder-Mead method.

References

[1] A. Jøsang, T. Bhuiyan, "Optimal Trust Network Analysis with Subjective Logic," in Proc. of 2nd SECURWARE 2008, Aug. 2008,

pp. 179-184. [Online]. Available: http://dx.doi.org/10.1109/SECURWARE.2008.64.

[2] S. Adali, R. Escriva, M. K. Goldberg, M. Hayvanovych, M. Magdon-Ismail, B. K. Szymanski, W. A. Wallace, G. Williams, "Measuring Behavioral Trust in Social Networks," in Proc. of IEEE ISI'2010, 2010, pp. 150-152. [Online]. Available: http://dx.doi.org/10.1109/ISI.2010.5484757.

[3] A. Jøsang, R. Hayward, S. Pope, "Trust Network Analysis with Subjective Logic," in Proc. of 29th ACSC2006, Jan. 2006, pp. 85-94.

[4] L. Ding, P. Kolari, S. Ganjugunte, T. Finin, A. Joshi, "Modeling and Evaluating Trust Network Inference," In Proc. of 7th AAMAS'2004, July 2004, pp. 21-32.

[5] P. Massa, P. Avesani, "Controversial users demand local trust metrics: an experimental study on Epinions.com community," in Proc. of 20th national conference on Artificial intelligence –Vol.1, 2005, pp. 121-126.

[6] L. Page, S. Brin, R. Motwani, T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford InfoLab Technical Report, SIDL-WP-1999-0120, 1999.

[7] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in Proc. of the 12th international conference on World Wide Web, 2003, pp. 640-651. [Online]. Available: http://dx.doi.org/10.1145/775152.775242.

[8] O. Savas, G. Jin, J. Deng, "Trust management in cloud-integrated Wireless Sensor Networks," In. Proc. of CTS 2013, 2013, pp.334-341. [Online]. Available: http://dx.doi.org/10.1109%2FCTS.2013.6567251.

[9] J. Lopez, R. Roman, I. Agudo, C. Fernandez-Gago, "Trust management systems for wireless sensor networks: Best practices," Computer Communications, Vol. 33, No. 9, pp. 1086–1093, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.02.006.

[10] A. Jøsang, V. A. Bondi, "Legal Reasoning with Subjective Logic," Artificial Intelligence and Law, Vol. 8, No. 4, pp. 289-315, 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1011219731903.

[11] A. Jøsang, D. McAnally, "Multiplication and Comultiplication of Beliefs," International Journal of Approximate Reasoning, Vol. 38, No. 1, pp. 19-51, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.ijar.2004.03.003.

[12] U. Maurer, "Modeling a Public-Key Infrastructure," in Proc. of ESORICS-LNCS Vol. 1136, 1996, pp. 325-350.

[13] A. Gutscher, "A Trust Model for an Open, Decentralized Reputation System," in Proc. IFIPTM, 2007, pp. 285-300. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-73655-6_19.

[14] T. Sun, M. K. Denko, "A Distributed Trust Management Scheme in the Pervasive Computing Environment," in Proc. of CCECE 2007., April 2007, pp. 1219–1222, [Online]. Available: http://dx.doi.org/10.1109/CCECE.2007.311.

[15] K. Sentz, S. Ferson, "Combination of Evidence in Dempster-Shafer Theory," SANDIA Tech. Report, SAND2002-0835, 2002.

[16] A. Gutscher, J. Heesen and O. Siemoneit, "Possibilities and Limitations of Modeling Trust and Reputation," in Proc. CEUR Workshop, 2008.

[17] K. Nordheimer, T. Schulze, D. Veit., "Trustworthiness in Networks: A Simulation Approach for Approx. Local Trust and Distrust Values," in Proc. of IFIP AICT, 2010, pp.157-171. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13446-3_11

[18] T. Bhuiyan, A. Jøsang, Y. Xu, "An analysis of trust transitivity taking base rate into account," in Proc. of Ubiquitous, Autonomic and Trusted Computing, 2009, pp. 34-39.

[19] P. Flocchini, F. L. Luccio, "Routing in Series Parallel Networks," Theory of Computing Systems, Vol. 36, pp. 137-157, 2003. [Online]. Available: http://dx.doi.org/10.1007/s00224-002-1033-y.

[20] C. T. Kelly, "Iterative Methods for Optimization (Frontiers in Applied Mathematics)," pp. 135-136, Society for Industrial and Applied Mathematics, 1st edition, Jan. 1987.

[21] D.F. Shanno, "Conditioning of quasi-Newton methods for function minimization," Mathematics of Computation, vol. 24, no. 111, pp. 647-657, 1970.

[22] R. Fletcher, C. M. Reeves, "Function minimization by conjugate gradients," Computer Journal, vol. 7, no. 2, pp. 148-154, 1964.

[23] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization," SIAM Journal on Scientific Computing, vol. 16, no. 5, pp. 1190-1208, 1995. Available: http://dx.doi.org/10.1137/0916069.

[24] A. R. Conn, K. Scheinberg, L. N. Vicente, "Introduction to Derivative-Free Optimization (Mps-Siam Series on Optimization)," pp. 113-120, Society for Industrial and Applied Mathematics, 1st edition, Jan. 2009.