OpinionWalk: An Efficient Solution to Massive Trust Assessment in Online Social Networks

***Abstract*—Massive trust assessment (MTA) in an Online Social Network (OSN), i.e., computing the trustworthiness of all users in the network, is crucial in various OSN-related applications. Existing solutions are either too slow or inaccurate in addressing the MTA problem. We propose the OpinionWalk algorithm that accurately and efficiently conducts MTA in an OSN. OpinionWalk models trust by the Dirichlet distribution and uses a matrix to represent the direct trust relations among users. From the perspective of a user, other users' trustworthiness are stored in a column vector that is iteratively updated when the algorithm "walks" through the network, in a breadth-first search manner. We identify the overlapping subproblems property in MTA and prove OpinionWalk is a more efficient solution. The accuracy and execution time of OpinionWalk are evaluated and compared to benchmark algorithms including EigenTrust, TrustRank, MoleTrust, TidalTrust and AssessTrust, using two real-world datasets (Advogato and Pretty Good Privacy). Experimental results indicate that OpinionWalk is an efficient and accurate solution to MTA, compared to previous algorithms.**

***Keywords*—Massive trust assessments, online social networks, three-valued subjective logic, trust model**

I. Introduction

A. Motivations and Problem Statement

An online social network (OSN) is a type of online services that facilitate the building of social relations/connections among people who have similar interests or backgrounds, and most importantly trust each other in real life. Trust is a hidden fabric in OSNs that enables reciprocal online interactions among users [1]. Besides fostering social interactions, the trust embedded in ONSs can also be used to improve various aspects of our life [2], [3]. For example, the Lending Club[1] company has leveraged the trust relations among users in Facebook.com[2] to improve its online peer-to-peer lending service. It offers a mechanism to evaluate the trustworthiness of potential borrowers, from the perspective of a lender. A fundamental research problem here is to identify the most trustworthy borrowers based on their social connections to the lender(s). To address this problem, we need a mechanism to conduct massive trust assessment (MTA) in OSNs, i.e., computing the trustworthiness of all users (e.g., potential borrowers), from the perspective of a user (e.g., the lender).

An OSN is usually modeled as a social graph where vertices represent users, and edges denote the trust relations among users. An edge in the graph typically reflects a user's opinion about the trustworthiness of another user. Therefore, in this paper, the terms *opinion* and *trust relation* are used interchangeably to refer to an edge. An opinion can be either a direct or an indirect opinion. While the former is designed for users who have direct interactions between each other, the later is applied to users who do not know each other in the network. If a user trusts another user, the first user who is regarded as the trustor can rely on the second, known as the trustee, to perform expected actions [4]. Therefore, *the MTA problem is to compute the trustworthiness of all trustees, from the point of view of the trustor, based on the trust social graph*.

[1]http://blog.lendingclub.com/facebook-and-lending-club-looks-like-its-working/

[2]http://www.techinsider.io/facebooks-patent-may-change-your-credit-2015-8

B. Limitations of Prior State of the Art

There are many algorithms applicable to massive trust assessment in OSNs, e.g., EigenTrust (ET) [5], TrustRank (TR) [6]), MoleTrust (MT) [7] and TidalTrust (TT) [8]. While EigenTrust and TrustRank focus on ranking users based on their relative trustworthiness, MoleTrust and TidalTrust are able to compute users' absolute trustworthiness. By searching a social graph with weighted averages, above-mentioned algorithms compute the trustworthiness of all trustees based on their social connections to the trustor. The time complexity of these algorithms is $O(n^2)$ where n is the number of nodes in the graph. One limitation of these algorithms is that they tend to give inaccurate assessments as trusts is modeled by a single real number.

To achieve more accurate trust assessment results, trust is modeled as statistical distributions in subjective logic (SL) [9] and three-valued subjective logic (3VSL) [4]. Based on 3VSL, the AssessTrust (AT) algorithm is proposed to accurately compute the trustworthiness between any two users in [4]. Although AT gives accurate trust assessment results, applying it to MTA yields a time complexity of $O(n^K)$ where K is the network's diameter. *In summary, existing algorithms are either inaccurate or very slow in solving the MTA problem.*

C. Proposed Approach

Based on 3VSL, we propose the OpinionWalk algorithm to efficiently and accurately address the MTA problem. We first use an *opinion matrix* to represent the topology of a trust social network. Each entry in the *opinion matrix* is a direct opinion of two corresponding users, which is modeled as a Dirichlet distribution based on 3VSL. We then design a set of matrix-like operations where traditional multiplication and summation operations are replaced by the discounting and combining operations. The discounting and combining operations are used to model trust propagation and fusion by 3VSL. Based on these newly defined operations, OpinionWalk starts from the trustor, searches the network, and compute users' trustworthiness in an iterative manner.

We prove that OpinionWalk is an equivalent implementation of AT and offers a better time complexity of $O(n^2)$ in

solving the MTA problem. We also compare the execution time and accuracy of OpinionWalk to existing algorithms, including AT, ET, MT, TR and TT, with two real-world datasets. Experimental results validate the accuracy and efficiency of OpinionWalk in solving the MTA problem.

D. Technical Challenges and Solutions

The MTA problem can be accurately solved by applying the AT algorithm n times, which yields an overall time complexity of $O(n^K)$. Therefore, the first technical challenge is to keep the accuracy of AT, at the same time, reduce the time complexity. To address this challenge, we design the OpinionWalk algorithm that searches the network in a BFS manner, and compute users' trustworthiness iteratively. To enable the iterative computations, we use an opinion matrix to represent the network and an individual opinion vector to store the trustworthiness of all users. When OpinionWalk "walks" through the network, the individual opinion vector is updated accordingly. When OpinionWalk finishes, the trustworthiness of all users are obtained. This design could reduce the time complexity from $O(n^K)$ to $O(n^2)$.

To enable BFS-based search in OpinionWalk, it is critical to ensure the overlapping subproblems property exists in MTA, if trust is modeled by the 3VSL model. In other words, to compute the trustworthiness of the trustor's $(k+1)$-hop friends, based on 3VSL, can we reuse the trustworthiness of his k-hop friends? This is a challenge because the k- and $(k+1)$-hop friends could be connected in various ways, resulting in different computations, according to 3VSL. We define the correct rules of applying k-hop friends' trustworthiness to compute the trustworthiness of $(k+1)$-hop friends, based on 3VSL. Therefore, OpinionWalk develops solutions to subproblems and reuse them to obtain the solution of the original MTA problem. The existence of overlapping subproblems property in MTA supports our adoption of BFS-based search in OpinionWalk.

The third technical challenge is to prove the correctness of OpinionWalk, i.e., it gives the same trust assessment results as AT, no matter what the network topology is. This is a challenge because these two algorithms are designed for different purposes and implemented in completely different ways. To address this challenge, we first prove the operations between opinion matrix and individual opinion vector equivalently implement the discounting and combining operations in simple network topologies. We then extend the proof into an arbitrary topology and recursively show that all sub MTA problems are equivalently solved by OpinionWalk and AT. Finally, we conclude that both OpinionWalk and AT provide the same MTA result.

E. Key Contributions

The key contributions of this paper are concluded as follows. First, an efficient 3VSL-based algorithm, OpinionWalk, is proposed to cope with the MTA problem. Second, the correctness of OpinionWalk is proved and its time complexity is analyzed. Third, trust assessment accuracy and the execution time of OpinionWalk are evaluated in real-world datasets and compared to benchmark algorithms.

The rest of this paper is organized as follows. In section II, we introduce the related work. In section III, we give problem statement and background knowledge of this paper. In section IV we introduce the OpinionWalk algorithm and analyze its time complexity, In section V, we prove the correctness of OpinionWalk. In section VI, we evaluate the runtime and accuracy of OpinionWalk. We conclude our work in section VII.

II. Related Work

Approaches to massive trust assessment in OSNs can be roughly divided into two broad categories, based upon how trust is modeled. Assuming trust is a real number, researchers studied how to compute relative trust [5], [6] and absolute trust [7], [8] in an OSN. On the other hand, trust can be modeled as a statistical distribution [4], [9]–[11], so more accurate trust assessments are realized.

In the first category, relative trust is first studied in peer-to-peer file-sharing networks where peers use trust values to choose trustworthy peers from which they download files [5]. The EigenTrust algorithm proposed in [5] starts from a peer and searches for trustworthy peers based on the following rules. It moves from a peer to another with the probability that is proportional to the other peer's trust score, i.e., higher the trust score, higher the moving probability. In this way, EigenTrust will more likely reach trustworthy peers than untrustworthy ones. Later on, the relative trust of web pages is investigated in [6] to identify spam pages. The TrustRank algorithm proposed in [6] again employs random walk on the network to rank the trustworthiness of web pages. Both EigenTrust and TrustRank can be viewed as a variant of the PageRank algorithm that is a well-known solution to assigning importance scores to pages on the Internet. These algorithms, however, only generate trust rankings instead of absolute trust values of peers/pages.

Based on the personal opinion of the trustor, for example, MoleTrust [7] proposes a method to compute the trustworthiness of a particular user in a personalized way. MoleTrust starts from the trustor and walks through the trust social network until the trustee is reached. While walking through the network, it only considers incoming edges with trust scores greater than 0.6 and ignores those with lower trust scores. Then, a user's trust score is computed by averaging all accepted incoming edges weighted by the trust scores of the users from whom the edges orientate. Similarly, TidalTrust [8] recursively searches the network with a weighted average approach. The difference between TidalTrust and MoleTrust is that TidalTrust considers only the path(s) with the highest trust score(s) between the truster and trustee, however, MoleTrust uses all paths as long as the trust score of each edge along the paths is greater than 0.6. Recently, the evolution or dynamics of trust in OSNs is studied in FluidRating [12]. FluidRating uses fluid dynamics theory to understand the evolution of trust in OSNs. Specifically, when a new opinion is received, a user refines his/her opinion through fluid exchanges with his/her neighbors.

In the second category, trust is modeled as a statistical distribution, e.g., in subjective logic [9], [10], CertProp [11] and three-valued subjective logic [4]. In this way, trust propagation and fusion are treat as the multiplication and summation

of statistical distributions. Comparing to solutions in the first category, these works achieve a higher accuracy in trust assessments. However, they have difficulty in handling complex networks due to the limitations identified in [4]. To enable trust assessment over large-scale networks, the AssessTrust algorithm is proposed in [4]. AssessTrust is based on the three-valued subjective logic model, and is proven to be able to handle arbitrary networks. A major limitation of AssessTrust is that it is designed to compute the trustworthiness for one user and thus is very slow and inefficient in solving the MTA problem. In summary, existing solutions either have trouble providing accurate trust assessments or are inefficient in solving the MTA problem in a large-scale OSN.

III. Problem Statement and Background

A. Problem Statement

A trust social network is modeled as a directed graph $G(V, E, \omega)$ where vertex $i \in V$ represents a user, and edge $e(i, j)$ denotes user i trusts user j. The weight on edge $e(i, j)$ indicates how user i trusts j. In this way, we can define the MTA problem as follows.

Given a trust social network $G(V, E, \omega)$, $\forall$ i and j, s.t. $i, j \in V$, $\exists$ at least one path from i to j, how to compute the trustworthiness of users $\{j \in V, j \neq i\}$, from user i's perspective.

The weight on an edge can be modeled as a real number or a statistical distribution, resulting in different accuracies in trust assessments. To achieve an accurate MTA, we rely on 3VSL to model the weight of an edge as a Dirichlet distribution.

B. Three-Valued Subjective Logic (3VSL)

Because the shape of a Dirichlet distribution is controlled by a set of parameters, we use these parameters to represent a Dirichlet distribution. Given trustor i and trustee j, i's opinion about the trustworthiness of j is a Dirichlet distribution denoted by a tuple

$$(b_{ij}, d_{ij}, n_{ij}, e_{ij})|a_{ij},$$

where $b_{ij} + d_{ij} + n_{ij} + e_{ij} = 1$.

In the definition, b_{ij}, d_{ij}, n_{ij} represent the probabilities that j will behave as expected, unexpected, neither expected or unexpected (i.e., uncertain), respectively. These values are determined by the numbers of evidences supporting the corresponding events. e_{ij} represents the prior uncertainty about whether j is trustworthy. Posterior uncertainty n_{ij} exists due to the fact that evidences are distorted when trust propagates over the network. a_{ij} is a constant number formed from impressions without solid evidences, e.g. prejudice and preference. As a_{ij} is not our focus, we omit it and model an opinion as a vector $(b_{ij}, d_{ij}, n_{ij}, e_{ij})$. We use ω_{ij} and Ω_{ij} to denote i's direct and indirect opinions on j, respectively. Note that both direct and indirect opinions are modeled in the same form, a Dirichlet distribution.

Based on 3VSL, we further define a special opinion that will be used to initialize the OpinionWalk algorithm.

Definition 1 (Uncertain Opinion): An uncertain opinion $\mathbb{O}$ is defined as

$$\mathbb{O} \triangleq (0, 0, 0, 1),$$

that indicates the trustor is totally uncertain about the trustee's trustworthiness.

To model trust propagation and fusion in OSNs, 3VSL defines the discounting and combining operations as follows. The discounting operation $\Delta(\omega_{is}, \omega_{sj})$ is used to compute i's opinion about j's trustworthiness based on s's opinion on j where s is a mutual friend of i and j. The discounting operation yields a new opinion Ω_{ij} where

$$\begin{cases} b_{ij} = b_{is} b_{sj} \\ d_{ij} = b_{is} d_{sj} \\ n_{ij} = 1 - b_{ij} - d_{ij} - e_{sj} \\ e_{ij} = e \end{cases}.$$

Here, we have $e = e_{sj}$ if $e_{is} \neq 1$; otherwise, $e = 1$. Note the direct opinions ω_{is} and ω_{sj} can be replaced by indirect opinions in the discounting operation.

The combining operation $\Theta(\omega'_{ij}, \omega''_{ij})$ is used to combine i's two opinions about j's trustworthiness. The combining operation will generate a new opinion where

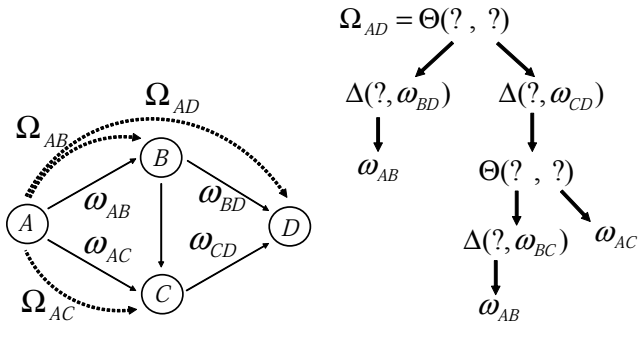
$$\begin{cases} b_{ij} = \dfrac{e''_{ij} b'_{ij} + e'_{ij} b''_{ij}}{e'_{ij} + e''_{ij} - e'_{ij} e''_{ij}} \\ d_{ij} = \dfrac{e''_{ij} d'_{ij} + e'_{ij} d''_{ij}}{e'_{ij} + e''_{ij} - e'_{ij} e''_{ij}} \\ n_{ij} = \dfrac{e''_{ij} n'_{ij} + e'_{ij} n''_{ij}}{e'_{ij} + e''_{ij} - e'_{ij} e''_{ij}} \\ e_{ij} = \dfrac{e'_{ij} e''_{ij}}{e'_{ij} + e''_{ij} - e'_{ij} e''_{ij}} \end{cases}.$$

Here, ω'_{ij} or ω''_{ij} could be replaced by i's indirect opinions on j as well. Based on the above definitions, we can easily prove the following corollaries.

Corollary 1: Applying the discounting operation on $\mathbb{O}$ and an opinion ω, we have $\Delta(\omega, \mathbb{O}) = \mathbb{O}$ and $\Delta(\mathbb{O}, \omega) = \mathbb{O}$.

Corollary 2: Applying the combining operation on $\mathbb{O}$ and an opinion ω, we have $\Theta(\omega, \mathbb{O}) = \Theta(\omega, \mathbb{O}) = \omega$.

C. AssessTrust (AT) Algorithm



(a) Bridge topology (b) Corresponding parsing tree

Fig. 1. An illustration of the AT algorithm based on the bridge topology.

In the following, we use an example shown in Fig. 1 to explain how the AssessTrust (AT) algorithm works. The AT

algorithm starts from the trustee, e.g., D in Fig. 1(a), searches the network backwards and recursively computes the trustworthiness of users one by one. As a result, we get a parsing tree, shown in Fig. 1(b), that describes how discounting and combining operations are applied in computing A's opinion on D. Traversing the parsing tree in a bottom-up manner, A's indirect opinion on D Ω_{AD} can be computed as

$$\Theta\left(\Delta(\omega_{AB},\omega_{BD}),\Delta(\Theta(\Delta(\omega_{AB},\omega_{BC}),\omega_{AC}),\omega_{CD})\right). \quad (1)$$

To understand how AT is applied on the graph, we use $AT^k(i,j)$ to denote that it is the k-th time that AT is called to compute user i's opinion on j. At the first time when AT is called, A's opinion on D is computed from

$$\Theta\left(\Delta(\Omega_{AB},\omega_{BD}),\Delta(\Omega_{AC},\omega_{CD})\right),$$

where Ω_{AB} and Ω_{AC} are A's indirect opinions on B and C, respectively. These two opinions will then be provided by $AT^{(2)}(A,B)$ and $AT^{(3)}(A,C)$, respectively. In $AT^{(3)}(A,C)$, AT computes A's opinion on C as

$$\Theta\left(\Delta(\Omega_{AB},\omega_{BC}),\omega_{AC}\right),$$

where Ω_{AB} are computed by $AT^{(4)}(A,B)$. Finally, A's opinion on D can be computed from Eq. 1. Overall, AT is called four times in this example: $AT^{(1)}(A,D)$, $AT^{(2)}(A,B)$, $AT^{(3)}(A,C)$, $AT^{(4)}(A,B)$. Note that the $AT(A,B)$ is called twice in this example.

IV. Design of OpinionWalk

OpinionWalk is essentially a BFS-based algorithm that addresses the MTA problem in a more efficient way. Given a trust social network, OpinionWalk represents it as an opinion matrix

$$M=\begin{bmatrix}\omega_{11} & \omega_{12} & ... & \omega_{1n}\\ \omega_{21} & \omega_{22} & \cdots & \cdots\\ \cdots & \cdots & \cdots & \cdots\\ \omega_{n1} & \cdots & \cdots & \omega_{nn}\end{bmatrix},$$

where element ω_{ij} $(i,j\le n)$ denotes i's direct opinion on j. If i has no direction opinion on j, we use $\mathbb{O}$ to substitute ω_{ij}.

From the view of the trustor, e.g., user i, its opinions about the trustworthiness of other users are stored in the individual opinion vector

$$Y_i^{(k)}=\left[\Omega_{i1}^{(k)},\Omega_{i2}^{(k)},\cdots,\Omega_{ij}^{(k)},\cdots,\Omega_{in}^{(k)}\right]^{\mathrm{T}},$$

where $\Omega_{ij}^{(k)}$ denotes user i's opinion on j after OpinionWalk "walks" k hops on the network. The individual opinion vector is initialized as

$$\Omega_{ij}^{(1)}=\begin{cases}\omega_{ij}, & \text{if user } i \text{ directly connects to } j\\ \mathbb{O}, & \text{otherwise}\end{cases}.$$

In this way, when OpinionWalk searches the network, the individual opinion vector is updated according to

$$Y_i^{(k)}=M^T\odot Y_i^{(k-1)}.$$

A. Operations in OpinionWalk

The operator $\odot$ "multiplies" M and $Y_i^{(k-1)}$ to yield an updated $Y_i^{(k)}$ as follows.

$$\begin{aligned}Y_i^{(k)}&=M^T\odot Y_i^{(k-1)}\\&=\begin{bmatrix}\Theta\left(\Delta(\Omega_{i1}^{(k-1)},\omega_{11}),\cdots,\Delta(\Omega_{in}^{(k-1)},\omega_{n1})\right),\\ \Theta\left(\Delta(\Omega_{i1}^{(k-1)},\omega_{12}),\cdots,\Delta(\Omega_{in}^{(k-1)},\omega_{n2})\right),\\ \cdots\\ \Theta\left(\Delta(\Omega_{i1}^{(k-1)},\omega_{1n}),\cdots,\Delta(\Omega_{in}^{(k-1)},\omega_{nn})\right)\end{bmatrix}\\&=\left[\Omega_{i1}^{(k)},\Omega_{i2}^{(k)},\cdots,\Omega_{ij}^{(k)},\cdots,\Omega_{in}^{(k)}\right]^{\mathrm{T}},\end{aligned}$$

where Θ and Δ are the combining and discounting operations, respectively. As shown in Fig. 2, the function of $\odot$ is analogous to multiplying a matrix and a vector. The difference lies in the summation and multiplication operations are replaced by the combining and discounting operations, respectively. Let's look at element $\Omega_{ij}^{(k)}$ in $Y_i^{(k)}$ where $i\neq j$. It is computed by "multiplying" vectors $[\omega_{1j},\omega_{2j},\cdots,\omega_{nj}]$ and $\left[\Omega_{i1}^{(k-1)},\Omega_{i2}^{(k-1)},\cdots,\Omega_{in}^{(k-1)}\right]^{\mathrm{T}}$.

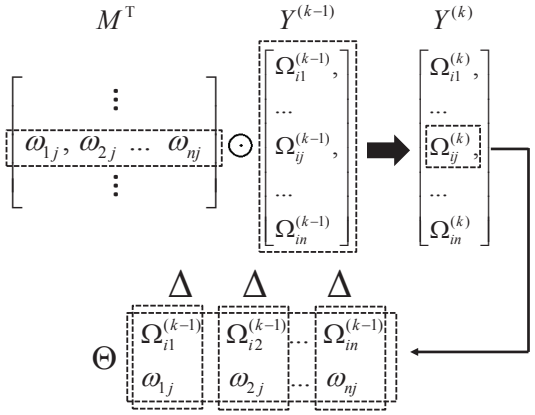


Fig. 2. A diagrammatic overview of the $\odot$ operation in OpinionWalk.

The principle of operator $\odot$ can be better explained by Fig. 3. In the figure, OpinionWalk starts from user i and searches the network level by level. Suppose OpinionWalk is currently searching the $(k-1)$-th level, and it finds a set of users that are $(k-1)$-hop away from i. Among these users, we assume m of them directly connect to user j, i.e., user j is k-hop away from i. We label these users as $s_1,s_2,\cdots,s_m$.

When OpinionWalk moves to the next level, i.e., the k-th level, it updates i's opinion on j as follows.

$$\Theta\left(\Delta(\Omega_{is_1}^{(k-1)},\omega_{s_1j}),\cdots,\Delta(\Omega_{is_m}^{(k-1)},\omega_{s_mj})\right)$$

It combines all m opinions that are computed from discounting ω_{s_lj} by Ω_{is_l}, for all possible $l=1,2,\cdots,m$. Due to Corollaries 1 and 2, this equation[3] can be further generalized to

$$\Theta\left(\Delta(\Omega_{i1}^{(k-1)},\omega_{1j}),\cdots,\Delta(\Omega_{in}^{(k-1)},\omega_{nj})\right). \quad (2)$$

[3]The equation here is generalized for presentation purposes. When it is implemented, only m opinions will be considered.

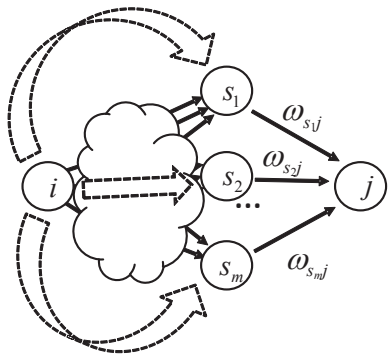


Fig. 3. Illustration of the principle of the $\odot$ operator in OpinionWalk.

It is possible that $\Omega_{ij}^{(k-1)} \neq \mathbb{O}$, i.e., OpinionWalk already got i's opinion on j from previous searches. In this case, i's opinion on j will be replaced by $\Omega_{ij}^{(k)}$. In other words, only the opinions $\Omega_{is}^{(k-1)}$ where $\omega_{sj} \neq \mathbb{O}$ are used in updating the individual opinion vector.

B. OpinionWalk Algorithm

Algorithm 1 OpinionWalk(G, i, H)

Require: A directed graph G with a trustor i and the maximum searching level H.
Ensure: i's opinion j where $j \neq i$.

```
1: Initialize M and Y_i^(1) based on G
2: k ← 1
3: while k < H do
4:    for all columns c_j ∈ M s.t. j ≠ i do
5:       for all direct opinions ω_sj ∈ c_j s.t. ω_sj ≠ 𝕆 do
6:          Ω_is^(k) ← Y_i^(k)[s]
7:          if Ω_is^(k) ≠ 𝕆 then
8:             if Ω_ij^(k+1) = 𝕆 then
9:                Ω_ij^(k+1) ← Δ(Ω_is^(k), ω_sj)
10:            else
11:               Ω_ij^(k+1) ← Θ(Ω_ij^(k+1), Δ(Ω_is^k, ω_sj))
12:            end if
13:         end if
14:      end for
15:      Y_i^(k+1)[j] ← Ω_ij^(k+1)
16:   end for
17:   k ← k + 1
18: end while
19: return Y_i^(k)
```

The pseudo-code of the OpinionWalk algorithm is shown in Algorithm 1. In the algorithm, line 3 controls how many hops OpinionWalk will search on the network. Lines 4-16 update the indirect opinion Ω_{ij} iteratively. Line 4 considers all users, other than i, as the trustees. Lines 5-14 combine all opinions derived from $\omega_{sj} \neq \mathbb{O}$. Line 6 obtains i's indirect opinion on one of its k-hop friends, e.g., s. If this opinion already exists, i discounts s's opinion on j to update $\Omega_{ij}^{(k+1)}$ in line 9. Otherwise, it checks another k-hop friend. Line 11 combines all opinions that are currently computed from $\omega_{sj} \neq \mathbb{O}$. Note that line 11 essentially combines opinions one by one, so $\Omega_{ij}^{(k+1)}$ equals to

$$\Theta(\Delta(\Omega_{i1}^{(k-1)}, \omega_{1j}), \cdots, \Theta(\Delta(\Omega_{in-1}^{(k-1)}, \omega_{n-1j}), \Delta(\Omega_{in}^{(k-1)}, \omega_{nj}))).$$

Because the combining operation is associative [4], the above equation is the same as Eq. 2. After processing all users connecting to j, at line 14, the newly computed Ω_{ij} is used to update the corresponding element in the individual opinion vector. When i's opinions on all possible j's are updated, at line 16, OpinionWalk searches the next level. Finally, the vector $Y_i^{(k)}$ will contain i's opinions about the trustworthiness of all other users.

C. Overlapping Subproblems Property

In the following, we will use the bridge topology in Fig. 1 to show the overlapping subproblems property in MTA when it is solved by OpinionWalk. Using the same example, we also show OpinionWalk is a more efficient algorithm, comparing to AT.

According to the OpinionWalk algorithm, user A's opinions on B, C and D are updated as follows. OpinionWalk starts from A and searches A's 1-hop neighbors. After this search, A's opinions on B and C are updated to

$$\Omega_{AB}^{(1)} = \omega_{AB} \text{ and } \Omega_{AC}^{(1)} = \omega_{AC}.$$

When OpinionWalk searches the next level, i.e., $k = 2$, A's opinion on B does not change but its opinion on C and D are changed to

$$\Omega_{AC}^{(2)} = \Theta\left(\Delta(\Omega_{AB}^{(1)}, \omega_{BC}), \omega_{AC}\right)$$

and

$$\Omega_{AD}^{(2)} = \Theta\left(\Delta(\Omega_{AB}^{(1)}, \omega_{BD}), \Delta(\Omega_{AC}^{(1)}, \omega_{CD})\right).$$

Inserting $\Omega_{AB}^{(1)} = \omega_{AB}$ and $\Omega_{AC}^{(1)} = \omega_{AC}$ into the above equations, we have

$$\Omega_{AC}^{(2)} = \Theta\left(\Delta(\omega_{AB}, \omega_{BC}), \omega_{AC}\right)$$

and

$$\Omega_{AD}^{(2)} = \Theta\left(\Delta(\omega_{AB}, \omega_{BD}), \Delta(\omega_{AC}, \omega_{CD})\right).$$

When OpinionWalk reaches the third level, i.e., $k = 3$, only A's opinion on D is updated to

$$\Omega_{AD}^{(3)} = \Theta\left(\Delta(\Omega_{AB}^{(2)}, \omega_{BD}), \Delta(\Omega_{AC}^{(2)}, \omega_{CD})\right).$$

Replacing $\Omega_{AB}^{(2)}$ and $\Omega_{AC}^{(2)}$ with the results obtained when $k = 2$, we have A's opinion on D as

$$\Theta\left(\Delta(\omega_{AB}, \omega_{BD}), \Delta(\Theta(\Delta(\omega_{AB}, \omega_{BC}), \omega_{AC}), \omega_{CD})\right).$$

This equation is exactly the same as Eq. 1 that is obtained by the AT algorithm. Note that the solutions to sub-problems are re-used in OpinionWalk; however, the same sub-problem may be solved over and over again in AT. For example, the trustworthiness between A and B are computed twice in $AT^{(2)}(A, B)$ and $AT^{(4)}(A, B)$. Leveraging the overlapping subproblems property, OpinionWalk yields a better time complexity than AT, which will be formally analyzed in the next subsection.

D. Time Complexity Analysis

In this section, we analyze the time complexities of OpinionWalk and AssessTrust, and find OpinionWalk yields a better time complexity.

AT is a recursive algorithm, and the recurrence equation of its time complexity is

$$T(n) = (n-1) \cdot T(n-1) + C(n-1),$$

where the first $(n-1)$ denotes the maximum number of users connecting to the trustee. If there are n users in the network, there are at most $(n-1)$ users connecting to the trustee. To find the trustworthiness of a user in the $(n-1)$ users, it takes $T(n-1)$. After that, the opinions derived from the $(n-1)$ users are combined, which takes $C(n-1)$. Expending the above equation, we obtain AT's time complexity as

$$O\left(\sum_{i=1}^{K-1} \frac{(n-1)!}{(n-1-i)!}\right) = O\left(n^{K-1}\right),$$

where K is the diameter of the network, measured by number of hops. To solve the MTA problem, AT needs to compute the trustworthiness of every user in the network, so its time complexity goes to

$$O\left((n-1) \cdot n^{K-1}\right) = O\left(n^K\right).$$

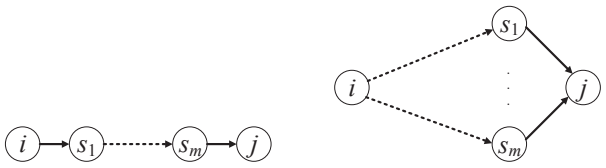
In the OpinionWalk algorithm, there are two nested loops: lines 4-16 and lines 5-14. The number of iterations in each loop is $O(n)$, so the time complexity of the nested loops is $O(n^2)$. The parameter H controls the maximum number of levels to be searched by OpinionWalk. As H is usually a constant number, OpinionWalk's time complexity becomes $O(n^2)$. In the extreme case where $H = K$, OpinionWalk's time complexity will be $O(Kn^2)$ that is still much better than AT's $O\left(n^K\right)$.

In addition, because OpinionWalk uses iterative procedures, it offers a faster execution time compared to the recursive implementation of AT. Because large number of stack operations are involved, AT consumes much more system memory. Faster execution time is extremely useful when the network gets larger and more complex, which are common in current OSNs.

V. Correctness of OpinionWalk

To prove OpinionWalk equivalently implements the AT algorithm, we first show that both AT and OpinionWalk generate the same result if the network topology is either serial or parallel. Then, we show this is true for arbitrary network topologies.

If we zoom into a trust social network, two edges can be connected in series if they are incident to a vertex of degree 2, or in parallel if they join the same pair of distinct vertices. Therefore, two users can be connected in a serial topology shown in Fig. 4(a), or a parallel topology shown in Fig. 4(b). Note that the paths from i to $s_1, s_2, \cdots, s_m$ in Fig. 4(b) are disjoint, i.e., no sharing edges along the paths.



(a) Serial topology (b) Parallel topology

Fig. 4. Illustrations of two fundamental topologies in an OSN.

A. Serial Network Topology

Lemma 1: Given two users i and j who are connected by m users in a serial topology, the opinion Ω_{ij} computed from AT and OpinionWalk will be the same.

Proof: We use $s_1, s_2, \cdots, s_m$ to denote the users that connects i to j, as shown in Fig. 4(a). According to the AT algorithm, i's trustworthiness of j is

$$\Omega_{ij} = \Delta\left(\omega_{is_1}, \Delta(\omega_{s_1s_2}, \cdots \Delta(\omega_{s_{m-1}s_m}, \omega_{s_mj}))\right). \quad (3)$$

In the OpinionWalk algorithm, the opinion matrix is

$$M = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \omega_{is_1} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \omega_{s_1s_2} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \omega_{s_mj} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

In the matrix, except for opinions $\omega_{is_1}, \omega_{s_1s_2}, \cdots, \omega_{s_mj}$, all other opinions are uncertain opinions $\mathbb{O}$. The initial individual opinion vector is

$$Y_i^{(1)} = [\mathbb{O}, \cdots, \omega_{is_1}, \cdots, \mathbb{O}]^{\mathrm{T}}.$$

If OpinionWalk searches the 2nd level of the network, the individual opinion vector is updated to

$$Y_i^{(2)} = [\mathbb{O}, \cdots, \omega_{is_1}, \cdots, \Delta(\omega_{is_1}, \omega_{s_1s_2}), \cdots, \mathbb{O}]^{\mathrm{T}}.$$

where $\Omega_{is_1}^{(2)} = \omega_{is_1}$ and $\Omega_{is_2}^{(2)} = \Delta(\omega_{is_1}, \omega_{s_1s_2})$ are i's current opinions on users s_1 and s_2, respectively.

When OpinionWalk searches the $(m+1)$-th level, it will reach user j, and $Y_i^{(m+1)}$ becomes

$$\begin{aligned} Y_i^{(m+1)} &= M^{\mathrm{T}} \odot Y_i^{(m)} \\ &= \left[\mathbb{O}, \cdots, \Omega_{is_m}^{(m)}, \cdots, \Delta(\Omega_{is_m}^{(m)}, \omega_{s_mj}), \cdots, \mathbb{O}\right]^{\mathrm{T}} \\ &= \left[\mathbb{O}, \cdots, \Omega_{is_m}^{(m+1)}, \cdots, \Omega_{ij}^{(m+1)}, \cdots, \mathbb{O}\right]^{\mathrm{T}}. \end{aligned}$$

If we expend $\Omega_{ij}^{(m+1)}$ in the above equation, we will get

$$\begin{aligned} \Omega_{ij}^{(m+1)} &= \Delta(\Omega_{is_m}^{(m)}, \omega_{s_mj}) \\ &= \Delta(\Delta(\Omega_{is_{m-1}}^{(m-1)}, \omega_{s_{m-1}s_m}), \omega_{s_mj}) \\ &= \Delta(\Delta(\Delta(\omega_{is_1}, \omega_{s_1s_2}), \cdots, \omega_{s_{m-1}s_m}), \omega_{s_mj}) \end{aligned} \quad (4)$$

Because the discount operation is associative [4], Eqs. 3 and 4 give the same result. Therefore, given a serial topology, OpinionWalk equivalently implements AT. ■

B. Parallel Network Topology

Lemma 2: Given two users i and j who are connected by m users in a parallel topology, the opinion Ω_{ij} computed from AT and OpinionWalk will be the same.

Proof: We use $s_1, s_2, \cdots, s_m$ to denote the m users based on their distances to i, i.e., s_m is the farthest away from i. As shown in Fig. 4(b), because the paths from i to $s_1, s_2, \cdots, s_m$ are disjointed, the AT algorithm computes i's opinion on j as

$$\Theta\left(\Delta(\Omega_{is_1}, \omega_{s_1 j}), \Delta(\Omega_{is_2}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}, \omega_{s_m j})\right). \quad (5)$$

According to the OpinionWalk algorithm, the opinion matrix will be

$$M = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \omega_{s_1 j} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \omega_{s_2 j} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \omega_{s_m j} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

Let's assume after $k_1, k_2, \cdots, k_m$ searches, the OpinionWalk algorithm reaches users $s_1, s_2, \cdots, s_m$, respectively. After k_1 searches, OpinionWalk obtains i's opinion on s_1 as $\Omega_{is_1}^{(k_1)}$. Here, $\Omega_{is_1}^{(k_1)}$ is equal to Ω_{is_1} computed from AT in Eq. 5, due to Lemma 1.

During the (k_1+1)-th search, OpinionWalk reaches user j, and it updates i's opinion on j to

$$\Omega_{ij}^{(k_1+1)} = \Delta(\Omega_{is_1}, \omega_{s_1 j}).$$

After k_2 searches, OpinionWalk gets i's opinion on s_2 as Ω_{is_2}. At the (k_2+1)-th search, OpinionWalk hits user j and updates i's opinion on j to

$$\Omega_{ij}^{(k_2+1)} = \Theta\left(\Omega_{ij}^{(k_2)}, \Delta(\Omega_{is_2}, \omega_{s_2 j})\right).$$

From (k_1+1)-th to k_2-th searches, opinion Ω_{ij} is not updated, so we have $\Omega_{ij}^{(k_2)} = \Omega_{ij}^{(k_1+1)}$. Therefore, we have

$$\Omega_{ij}^{(k_2+1)} = \Theta\left(\Delta(\Omega_{is_1}, \omega_{s_1 j}), \Delta(\Omega_{is_2}, \omega_{s_2 j})\right).$$

Similarly, after $k_m + 1$ searches, i's opinion on j is updated to

$$\Theta\left(\Delta(\Omega_{is_1}^{(k_m)}, \omega_{s_1 j}), \Delta(\Omega_{is_2}^{(k_m)}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}^{(k_m)}, \omega_{s_m j})\right).$$

For every $l = 1, 2, \cdots, m$, we have $\Omega_{is_l}^{(k_l)} = \Omega_{is_l}^{(k_m)}$ as it does not change after the k_l-th search. Therefore, the above equation becomes

$$\Theta\left(\Delta(\Omega_{is_1}^{(k_1)}, \omega_{s_1 j}), \Delta(\Omega_{is_2}^{(k_2)}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}^{(k_m)}, \omega_{s_m j})\right).$$

For any $l = 1, 2, \cdots, m$, we know that $\Omega_{is_l}^{(k_l)}$ is equal to Ω_{is_l} computed by AT, due to Lemma 1. Therefore, OpinionWalk computes i's opinion on j as

$$\Theta\left(\Delta(\Omega_{is_1}, \omega_{s_1 j}), \Delta(\Omega_{is_2}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}, \omega_{s_m j})\right). \quad (6)$$

Because Eqs. 5 and 6 give the same result, we conclude that OpinionWalk equivalently implements AT on a parallel topology. ■

C. Arbitrary Topology

Given an arbitrary network topology shown in Fig. 5, we assume $m > 1$ nodes $\{s_1, s_2, \cdots, s_m\}$ directly connects to j. We use $k_1, k_2, \cdots, k_m$ to denote the last times when users $s_1, s_2, \cdots, s_m$ are visited by the OpinionWalk algorithm, respectively. Let $\{t_1, t_2, \cdots, t_r\}$ denote the users who are connected from i via either serial or parallel topologies.

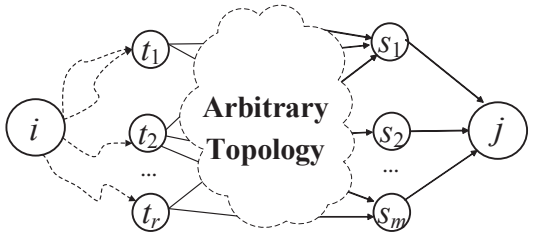


Fig. 5. Illustration of social network with an arbitrary topology.

Theorem 1: OpinionWalk equivalently implements the AT algorithm in an arbitrary network topology.

Proof: We prove the theorem in a recursive manner, i.e., reducing the original network into sub-network(s) and keep reducing the sub-network(s) until the base case is reached, i.e., two users are connected via either a serial or parallel topology.

Reduction rules

Case 1: There is only one user connecting to j, i.e., $m = 1$. In this case, according to the AT algorithm, Ω_{ij} is computed as

$$\Delta(\Omega_{is_1}, \omega_{s_1 j})$$

where Ω_{is_1} denotes i's opinion on s_1, and it can be computed by recursively calling the AT algorithm. For OpinionWalk, at the (k_1+1)-th search, it reaches j and updates i's opinion on j as

$$\Delta(\Omega_{is_1}^{(k_1)}, \omega_{s_1 j}).$$

Case 2: There are more than one user connecting to j, i.e., $m > 1$. In this case, AT computes Ω_{ij} as

$$\Theta(\Delta(\Omega_{is_1}, \omega_{s_1 j}), \Delta(\Omega_{is_2}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}, \omega_{s_m j})).$$

After $\hat{k}+1$ searches where $\hat{k} = \max(k_1, k_2, \cdots, k_m)$, OpinionWalk updates i's opinion on j to

$$\Theta(\Delta(\Omega_{is_1}^{(\hat{k})}, \omega_{s_1 j}), \Delta(\Omega_{is_2}^{(\hat{k})}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}^{(\hat{k})}, \omega_{s_m j})).$$

For any user s_l, where $l = 1, 2, \cdots, m$, because k_l is the last time that s_l was visited by OpinionWalk, Ω_{is_l} was not updated after the k_l-th search. Therefore, the above equation can be rewritten as

$$\Theta(\Delta(\Omega_{is_1}^{(k_1)}, \omega_{s_1 j}), \Delta(\Omega_{is_2}^{(k_2)}, \omega_{s_2 j}), \cdots, \Delta(\Omega_{is_m}^{(k_m)}, \omega_{s_m j})).$$

Summarizing the above two cases, we know that if $\Omega_{is_l} = \Omega_{is_l}^{(k_l)}$ for every $l = 1, 2, \cdots, m$, then OpinionWalk and AT yield the same result of Ω_{ij}. For any user s_l, Ω_{is_l} and $\Omega_{is_l}^{(k_l)}$ actually denote i's opinion on s_l computed by AT

and OpinionWalk, respectively. Both AT and OpinionWalk will work on the same sub-network $G' = G - e(s_l, j)$ that connects i to s_l. The sub-network G' can be further reduced by removing the edge connecting to s_l. Continuing this process, the original network G will eventually be reduced to the following base case.

Base Case

In the base case, user i connects to users $t_1, t_2, \cdots, t_r$ via either a serial or parallel topology. Based on Lemmas 1 and 2, AT and OpinionWalk give the same values of $\Omega_{it_1}, \Omega_{it_2}, \cdots, \Omega_{it_r}$. Overall, we prove that AT and OpinionWalk give the same result. ■

VI. Evaluation

We compare the performance of OpinionWalk (OW) to a few benchmark algorithms, including EigenTrust (ET), TrustRank (TR), MoleTrust (MT), TidalTrust (TT) and AssessTrust (AT), in two real-world datasets, Advogato [13] and Pretty Good Privacy (PGP) [11].

Advogato is obtained from an online software development community where an edge from a user to another user represents the user's opinion about the other's ability in software development. The trustworthiness between users are categorized into four levels. PGP is collected from a public key certification network where an edge from a user to another user indicates that the user certificates the trustworthiness of the other user. The trustworthiness between users are also classified into four categories. Statistics of these datasets are summarized in TABLE I.

TABLE I. Statistics of the Advogato and PGP Datasets

Dataset	# of Vertices	# of Edges	Avg Deg	Diameter
Advogato	6,541	51,127	19.2	4.82
PGP	38,546	31,7979	16.5	7.7

TABLE II. Transforming Ordinal Values to Real Numbers

Trust Level	Trust Opinion	Trust Value
1	(0.09, 0.81, 0, 0.1)	0.09
2	(0.36, 0.54, 0, 0.1)	0.36
3	(0.63, 0.27, 0, 0.1)	0.63
4	(0.9, 0, 0, 0.1)	0.9

Since the edge weights in Advogato and PGP are all ordinal data, we use the linear transformation technique [8], [11] to convert ordinal values into real numbers. As shown in TABLE II, trust data from original datasets are transformed to opinion vectors and real numbers, ranging from 0 to 1. All of these algorithms are implemented by python 2.7. The computer used for evaluations has an Intel CORE i7-3770 3.4 GHz CPU, 16G memory and 256G SSD hard drive.

A. Accuracy

The goal of this subsection is to check if OpinionWalk offers more accurate trust assessment results. MoleTrust and TidalTrust output trust values, EigentTrust and TrustRank give trust rankings; therefore, we divide the experiments into two groups. In the first group, we compare OpinionWalk to MoleTrust and TidalTrust by looking at the errors between computed trust values and the ground truth. In the second group, we compare OpinionWalk to EigenTrust and TrustRank by checking the Kendall's tau ranking correlation coefficient between the rankings generated by the algorithms and the ground truth. The AssessTrust algorithm is not compared here because it provides the same results as OpinionWalk, which is proved previously.

The experiments are carried out as follows. First, we randomly select a pair of users u and v where u connects to v in the datasets. We treat the direct opinion from u to v as the ground truth. Then, we remove the edge (u, v) from the datasets. We run the above-mentioned algorithms to compute v's trustworthiness, from u's point of view. Finally, we compare the computed results to the ground truth.

To make a fair comparison, we use the expected beliefs of computed opinions [4] to represent the trust values computed by OpinionWalk. In the first group, we use the mean absolute percentage error (MAPE) to measure the errors in computed trust values. In the second group, we rank users based on their trust values obtained from these algorithms. The ranking errors are then measured by the Kendall's tau ranking correlation coefficients between computed results and the ground truth. We repeat every experiment 100 times to get statistically significant results.

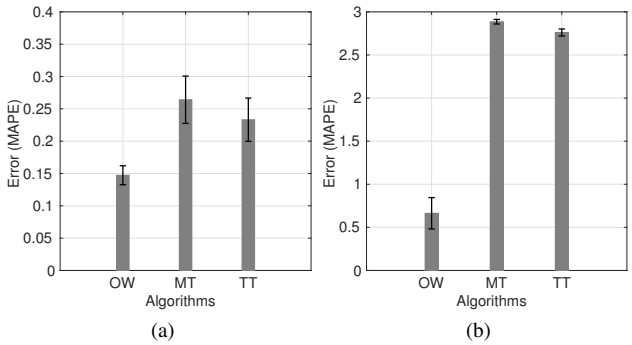


Fig. 6. Errors in trust assessments of different algorithms, using the (a) Advogato and (b) PGP datasets.

In Fig. 6, we find OpinionWalk achieves a higher accuracy, compared to MoleTrust and Tidal Trust, in both datasets. In Advogato, the MAPEs of OpinionWalk, MoleTrust and TidalTrust are 0.15, 0.26, 0.23, respectively. In PGP, the MAPEs of OpinionWalk, MoleTrust and TidalTrust are 0.77, 2.89 and 2.76, respectively. OpinionWalk outperforms other algorithms by 30% to 70% in different datasets. This is because OpinionWalk uses 3VSL to model trust, and 3VSL is believed to be the most accurate model for trust assessments in OSNs.

In Fig. 7, OpinionWalk also gives more accurate ranking results, compared to other algorithms. With the Advogato dataset, the Kendall's tau correlation coefficients between the ranking results generated from OpinionWalk and the ground truth are always greater than 0. Nearly 20% of ranking results are exactly the same as the ground truth, i.e., coefficients equal to 1. Using the PGP dataset, the Kendall's tau correlation coefficients of OpinionWalk are greater than 0.2. Moreover, about 40% of ranking results are the same as the ground truth. We clearly see that OpinionWalk offers more accurate ranking results than EigenTrust and RankTrust. From Figs. 6 and 7, we conclude that OpinionWalk gives more accurate

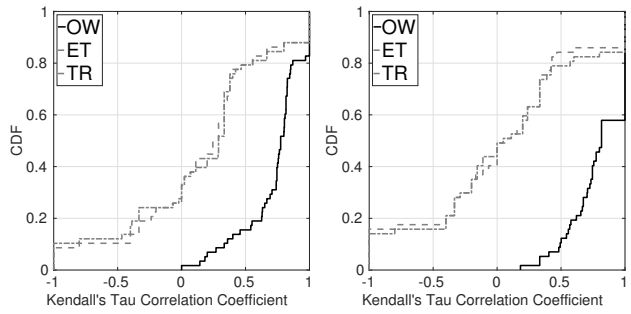


Fig. 7. The CDF of Kendall's tau ranking correlation coefficients of different algorithms, using the (a) Advogato and (b) PGP datasets.

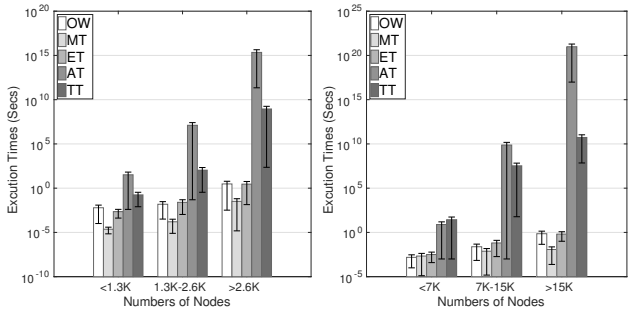


Fig. 8. Execution times of different algorithms, using the (a) Advogato and (b) PGP datasets.

trust assessments than MoleTrust, TidalTrust, EigenTrust, and RankTrust.

B. Execution Time

In the following, we will evaluate the runtimes of OpinionWalk, EigenTrust, TidalTrust, MoleTrust and AssessTrust. We do not repeat the evaluation of TrustRank here because its runtime is similar to EigenTrust's.

The experiments are conducted as follows. First, we randomly select a user, as the trustor, from each dataset. From the trustor, we identify a few subgraphs that contains the trustor's 1-hop, 2-hop and 3-hop friends, respectively. Then, we run the above-mentioned algorithms on each subgraph to compute the trustworthiness of all users, from the trustor's point of view. Based on the number of users, we divide experimental results into three groups. Finally, we plot the execution times with respect to the numbers of users in Fig. 8.

As shown in Fig. 8, TidalTrust and AssessTrust run much slower than other algorithms. The reason is that both TidalTrust and AssessTrust are executed $n-1$ times to solve the MTA problem. AssessTrust is the slowest algorithm because it is a recursive algorithm and it keeps re-solving the same subproblems over and over again. The runtimes of OpinionWalk, EigenTrust and MoleTrust are on the same order of magnitude, i.e., $O(n^2)$. Because discounting and combining operations are more complicated than multiplication and summation operations, OpinionWalk is slightly slower than MoleTrust. However, the execution time of OpinionWalk is similar to the second best solution, the EigenTrust algorithm. After evaluating the accuracies and execution times of different algorithms, we conclude that OpinionWalk is a more efficient and accurate solution to the MTA problem.

VII. Conclusion

OpinionWalk can be considered a unification taking the advantages of both AssessTrust and MoleTrust, i.e., it leverages the 3VSL model (to gain accuracy improvement) and searches networks in a BFS manner (for efficiency improvement). We proved that OpinionWalk is an equivalent implementation of the AssessTrust algorithm, yet offers a better time complexity of $O(n^2)$, compared to AssessTrust's $O(n^K)$. Experiments are conducted with two real-world datasets and results indicate OpinionWalk provides more accurate trust assessments. Due to the rapid development of OSN-related applications and services, OpinionWalk will attract more research interests in the future.

References

[1] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *ACM Computer Survey*, vol. 49, no. 1, pp. 10:1–10:35, May 2016.

[2] M. Sirivianos, K. Kim, and X. Yang, "Socialfilter: Introducing social trust to collaborative spam mitigation," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2300–2308.

[3] A. Mohaisen, N. Hopper, and Y. Kim, "Keep your friends close: Incorporating trust into social network-based sybil defenses," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1943–1951.

[4] G. Liu, Q. Yang, H. Wang, X. Lin, and M. Wittie, "Assessment of multi-hop interpersonal trust in social networks by three-valued subjective logic," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 1698–1706.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th International Conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 640–651.

[6] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB '04*, 2004, pp. 576–587.

[7] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions. com community," in *Proceedings of the National Conference on artificial Intelligence*, vol. 20, no. 1, 2005, p. 121.

[8] J. Golbeck, J. Hendler *et al.*, "Filmtrust: Movie recommendations using trust in web-based social networks," in *IEEE CCNC*, 2006, pp. 282–286.

[9] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference*, 2006, pp. 85–94.

[10] A. Josang, "A logic for uncertain probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 09, no. 03, pp. 279–311, 2001.

[11] C.-W. Hang, Y. Wang, and M. P. Singh, "Operators for propagating trust and their evaluation in social networks," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 1025–1032.

[12] W. Jiang, J. Wu, G. Wang, and H. Zheng, "Fluidrating: A time-evolving rating scheme in trust-based recommendation systems using fluid dynamics," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 1707–1715.

[13] P. Massa, M. Salvetti, and D. Tomasoni, "Bowling alone and trust decline in social network sites," in *Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on*, Dec 2009, pp. 658–663.