Trust Prediction via Belief Propagation

RICHONG ZHANG, State Key Laboratory of Software Development Environment, Beihang University
YONGYI MAO, University of Ottawa

The prediction of trust relationships in social networks plays an important role in the analytics of the networks. Although various link prediction algorithms for general networks may be adapted for this purpose, the recent notion of "trust propagation" has been shown to effectively capture the trust-formation mechanisms and resulted in an effective prediction algorithm. This article builds on the concept of trust propagation and presents a probabilistic trust propagation model. Our model exploits the modern framework of probabilistic graphical models, more specifically, factor graphs. Under this model, the trust prediction problem can be formulated as a statistical inference problem and we derive the belief propagation algorithm as a solver for trust prediction. The model and algorithm are tested using datasets from Epinions and Ciao, by which performance advantages over the previous algorithms are demonstrated.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems—*Human information processing*; H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.4.0 [**Information Systems Applications**]: General

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Trust propagation, social network, link prediction

ACM Reference Format:
Richong Zhang and Yongyi Mao. 2014. Trust prediction via belief propagation. ACM Trans. Inf. Syst. 32, 3, Article 15 (June 2014), 27 pages.
DOI: http://dx.doi.org/10.1145/2629530

1. INTRODUCTION

The rapid development of social networks has made them an enormous pool of information and an important platform for social interaction. The quality of user-generated contents in the social networks however varies significantly, which may depend on the intention, attitude, background, expertise, personality, and many other attributes of the user posting the contents. Users fetching information from a social network obviously desire only the valuable and trustworthy content. It is commonly recognized that the trust-worthiness of contents is usually positively correlated with the trust-worthiness of the user publishing it. Exploiting this understanding, many social networks, particularly those involving e-commerce and financial transactions, allow users to cast trusts on other users. Trust, as a means of social interaction, results in a complex network of trust relationships and has attracted great research interest in the recent years (see, for example, [Aberer and Despotovic 2001; Jamali and Ester 2009; Massa and Avesani

This work is supported partly by the National Natural Science Foundation of China (No. 61300070), partly by China 863 program (No. 2013AA01A213), China 973 program (No. 2014CB340305), and partly by the State Key Lab for Software Development Environment (SKLSDE-2013ZX-16).
Author's addresses: R. Zhang, State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, 37 Xueyuan Road, Beijing 100191, China; email: zhangrc@act.buaa.edu.cn; Y. Mao, School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Avenue, Ottawa, Ontario, K1N6N5; email: yymao@eecs.uottawa.ca.

DOI: http://dx.doi.org/10.1145/2629530

2007; Tang et al. 2012b; Tang et al. 2012a; Ma et al. 2009; Liu et al. 2011; Ziegler and Golbeck 2007; Liu et al. 2010; Golbeck 2009; Xiang et al. 2010; Liu et al. 2012; Leskovec et al. 2010]).

The study of trust relationship has pervaded many application domains, from peer-to-peer information systems [Aberer and Despotovic 2001], to electronic commerce [Jamali and Ester 2009], and to recommender systems [Massa and Avesani 2007], just to name a few. As one example, in [Massa and Avesani 2007], trust information is utilized in the design of recommender systems, in which a "trust-awareness" function is implemented. Massa and Avesani [2007] show that utilizing trust information allows the recommender system to provide product recommendations with improved quality.

This article deals with the "trust prediction" problem, the objective of which is to predict the possible trust relationships that are not yet observed in the network. Solving such a problem provides various benefits. For example, it helps to reduce the potential risks involved in the interaction of anonymous users; it allows a recommender system to utilize such information in product recommendation; it allows the network provider to promote user interactions and drive network dynamics in a healthy manner.

Earlier versions of this problem appear in the form of "link prediction" [Jeh and Widom 2002; Adamic and Adar 2003; Newman 2001; Liben-Nowell and Kleinberg 2007; Katz 1953; Liben-Nowell and Kleinberg 2003]. In general, link prediction aims at predicting new links that may potentially emerge in a network, and its context is not limited to trust networks. Nevertheless, previous algorithms developed for link prediction are readily applicable to trust prediction. These algorithms typically exploit either the neighbourhood structure of individual nodes or the path properties between pairs of nodes. For a comprehensive review of link prediction algorithms, the reader is referred to Liben-Nowell and Kleinberg [2007]. For signed networks, Leskovec et al. [2010] crafted a link prediction algorithm specifically for inferring the "signs" of network links. The authors however pointed out that their algorithm is not suitable for unsigned networks.

Despite the applicability and effectiveness of the link prediction algorithms, the work of Guha et al. [2004] remains as a landmark in trust-prediction literature. In Guha et al. [2004], the notion of "trust/distrust propagation" was for the first time formulated, based on which a trust and distrust prediction algorithm was proposed. The notion of trust/distrust propagation of Guha et al. [2004] includes four atomic propagation mechanisms and the evolution of a trust network is hypothesized to be driven by these mechanisms. The validity of these hypotheses is demonstrated experimentally by the effectiveness of a trust prediction algorithm built upon these hypotheses. A notion of trust propagation was also investigated in Ziegler and Lausen [2004]. In Ziegler and Lausen [2004], a propagation model is proposed for trust networks and the impact of the model parameters on network statistics is studied. It is worth pointing out, however, that the model of Ziegler and Lausen [2004] is not proposed for trust prediction purpose.

This article is a follow-up to Guha et al. [2004] and builds upon the trust propagation model therein. With a careful investigation of the four atomic propagation mechanisms, we first show that the set of atomic mechanisms [Guha et al. 2004] can be reduced to a set of two mechanisms only and the other two mechanisms can in fact be induced by this reduced set of postulates. The important consequence of this simplification is a reduction of modelling parameters in our model construction without sacrificing its expressive power. We then present a probabilistic version of this reduced set of mechanisms in order to accommodate the intrinsic noise, uncertainty and additional trust-formation mechanisms that can not be modelled as trust propagation. These hypotheses allow us to subsequently construct a factorized probability model, readily expressed in the framework of factor graphs [Kschischang et al. 2001]. Under our model, the trust prediction problem is translated to a statistical inference problem, and a well-principled powerful algorithm, the belief propagation algorithm, is developed as an algorithmic solver. Using datasets from Epinion[1] and Ciao[2], we demonstrate that our model and the belief propagation algorithm outperform various existing link prediction algorithms and as well the trust propagation algorithm of Guha et al. [2004].

To the best of our knowledge, this work is the first to formulate trust prediction as an inference problem in a probabilistic framework, and the first declarative approach (as opposed to the other procedural approaches) to this problem. There have been previous works utilizing random-walk models for link prediction [Liben-Nowell and Kleinberg 2007; Backstrom and Leskovec 2011], which borrows ideas from page ranking [Brin and Page 1998]. Such models formulates the link prediction problem as finding a stationary distribution of a homogeneous Markov chain and develops an algorithm based on an iterative procedure in positive matrix theory for finding the Perron vector of a positive matrix [Horn and Johnson 1990]. Such models, although probabilistic in nature, differ significantly from our model in both assumptions and methodology.

The remainder of this article is organized as follows. In Section 2, we first precisely state the trust prediction problem and present a concise review of the dominant existing algorithms (all of which are compared against in this work). In Section 3, we prove that the four atomic trust propagation mechanisms of Guha et al. [2004] can be reduced and present probabilistic hypotheses for the reduced set of mechanisms. These hypotheses are used for the construction of our probabilistic trust propagation model in Section 4. Section 5 derives the belief propagation algorithm for the proposed model, with certain details moved to Appendix (Section 9). Complexity issues of the algorithm are studied and handled in Section 6. Section 7 is dedicated to experimental study where our experimental procedures, various considerations and experimental results are presented. The article is briefly concluded in Section 8.

2. THE TRUST PREDICTION PROBLEM

Let directed graph $\mathcal{N}$ represent the observed trust network. That is, each vertex of $\mathcal{N}$ represents a user in the social network of interest, and an edge from vertex u to vertex v indicates that user u has expressed "trust" on user v. It is folk wisdom that the observed network $\mathcal{N}$ does not capture all pairs of users (a, b) in which a will eventually trust b: as the social network evolves, it is expected that more trust relationships, namely, directed edges, will be established. The trust prediction problem can then be stated as predicting the new edges that may arise in $\mathcal{N}$ as the network evolves.

When the problem is augmented with additional information of the social network, various techniques were developed for trust prediction (see, e.g., [Tang et al. 2012a, 2012b]). When the input of the problem contains only the observed network $\mathcal{N}$, which is the setup of this work, all approaches to this problem rely on certain hypotheses relating the appearance of new edges to the structure of the network $\mathcal{N}$. The hypotheses that have led to a reasonable prediction performance include the following[3].

—In Liben-Nowell and Kleinberg [2003], it is hypothesized that the likelihood that there is an edge from u to v is proportional to the number of common neighbours of u and v. This leads to a simple counting algorithm which we refer to as the Common Neighbour (CN) Algorithm.

—In Adamic and Adar [2003], a similarity measure of two users is defined using the Jaccard Similarity Coefficient [Levandowsky and Winter 1971] and it is hypothesized

[1]www.epinions.com.

[2]www.ciao.co.uk.

[3]We note that some of these hypotheses and algorithms were presented for the similar "link prediction" problem and not specifically developed for trust prediction.

that the likelihood that two users form a trust pair is proportional to their similarity. We refer to this algorithm as the Adamic-Adar (AA) Algorithm.

—The work of Katz [1953] in essence introduced a notion of "influence" of node u on node v for every path from u to v, which is assumed to decay exponentially with the path length. It was then hypothesized that the likelihood that u trusts v is proportional to the total influence of u on v along all paths from u to v. We call this algorithm the KZ Algorithm.

In Guha et al. [2004], a novel kind of hypothesis was introduced, where the authors postulated that there are four "atomic" mechanisms by which the trust (and distrust) relationships in the network may *propagate*. This perspective has resulted in the best known algorithm to date, which we refer to as the Trust Propagation (TP) algorithm in this article. The success of TP suggests that the postulated atomic propagation mechanisms serve as a good explanation for the emergence of new trust relationships in social networks and that the concept of "propagation" indeed plays an important role in the evolution of the trust network.

3. FROM TRUST PROPAGATION TO PROBABILISTIC TRUST PROPAGATION

3.1. Trust Propagation

This work is a follow-up on the notion of "trust propagation" in [Guha et al. 2004], for which we now present a concise review.

First recall the definitions of the four atomic propagation mechanisms: transposition, forwarding (referred to as direct propagation in Guha et al. [2004]), cocitation, and coupling:

Definition 1 (*Transposition*). If u trusts v, then v trusts u.

Definition 2 (*Forwarding*). If u trusts v and in addition v trusts w, then u trusts w.

Definition 3 (*co-citation*). If u_1 trusts both v_1 and v_2, and in addition u_2 trusts v_2, then u_2 trusts v_1.

Definition 4 (*coupling*). If u and v both trust w, and in addition x trusts u, then x trusts v.

For each of the four propagation mechanisms, TP constructs a propagation matrix based on the observed network $\mathcal{N}$. The four propagation matrices are then weighted and combined to give rise to an overall propagation matrix. This matrix is subsequently used as a linear transformation kernel and iteratively applied to the adjacency matrix of $\mathcal{N}$ (and in an extension, an aggregated version of the adjacency matrix). Despite the good performance of TP, it is arguable that TP is largely heuristic and procedural. As such, it is difficult to associate the procedures of TP with a quantitative interpretation. This weakness in physical interpretation also makes the linear-transformation approach taken in TP appear artificial rather than principled. In addition, the heuristic nature of the algorithm complicates the parameter selection in TP.

Another challenge faced by TP is its over-confidence about the role of atomic propagations in the evolution of trust networks. On one hand, TP assumes that the four atomic mechanisms are the only explanation for new edge formation in the network. This is evidently not the case in real social networks as there are other reasons for a new edge to emerge beyond the propagation of trust. On the other hand, TP insists that the four atomic mechanisms all occur deterministically in the network. Although the weights applied to the propagation matrices to an extent quantify the relative effect of each atomic mechanism, the weights applied globally to the matrices are conceivably

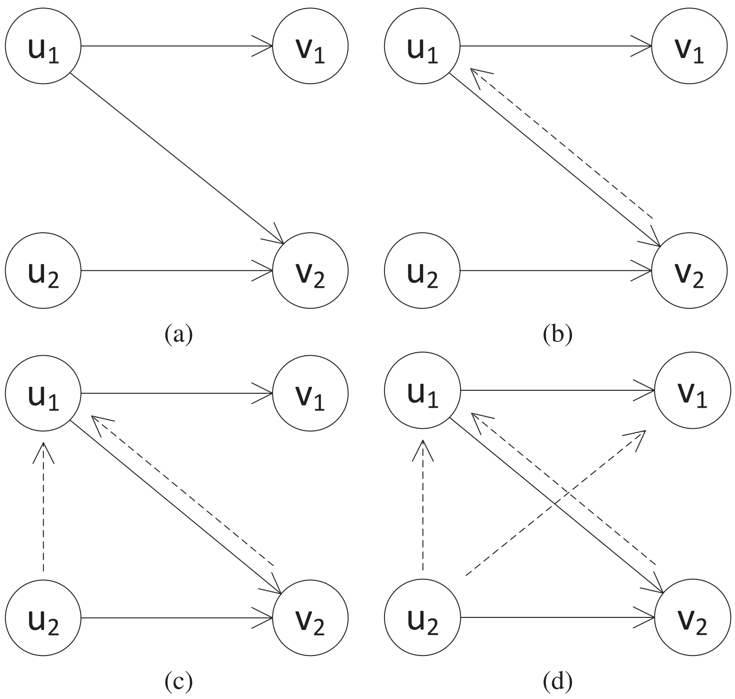


Fig. 1. Cocitation may result from transposition and forwarding.

insufficient to capture the varying "strength" of a propagation mechanism which may depend on the local structures the network.

In order to account for the limitation of the TP algorithm, this work develops a simple and well-principled model which has a sound physical interpretation and allows for an effective trust prediction algorithm.

3.2. Reduction of Atomic Propagation Mechanisms

The Occam's Razor principle advocates avoiding complex models until it becomes necessary. Interestingly, contemplating at the four atomic propagation mechanisms of Guha et al. [2004] leads to the following observation.

LEMMA 1. *Transposition and forwarding can induce both cocitation and coupling.*

The proof of this lemma can be visualized graphically as in Figures 1 and 2, which we now explain.

PROOF. First we show that cocitation may result from forwarding and transposition (Figure 1). Suppose that the observed network is the graph in (a). Since there is an edge $u_1 \to v_2$, there is an edge $v_2 \to u_1$ by transposition (b). Since both edges $u_2 \to v_2$ and $u_2 \to v_1$ exist, edge $u_2 \to u_1$ also exists by forwarding (c). Finally since both edges $u_2 \to u_1$ and $u_1 \to v_1$ exist, edge $u_2 \to v_1$ also exists by forwarding (d). This proves that cocitation may result from forwarding and transposition.

Now we show that coupling may also result from transposition and forwarding (Figure 2). Suppose that the observed network is the graph in (a). Since there is an edge $v \to w$, there is also an edge $w \to v$ by transposition (b). Since both edges $u \to w$

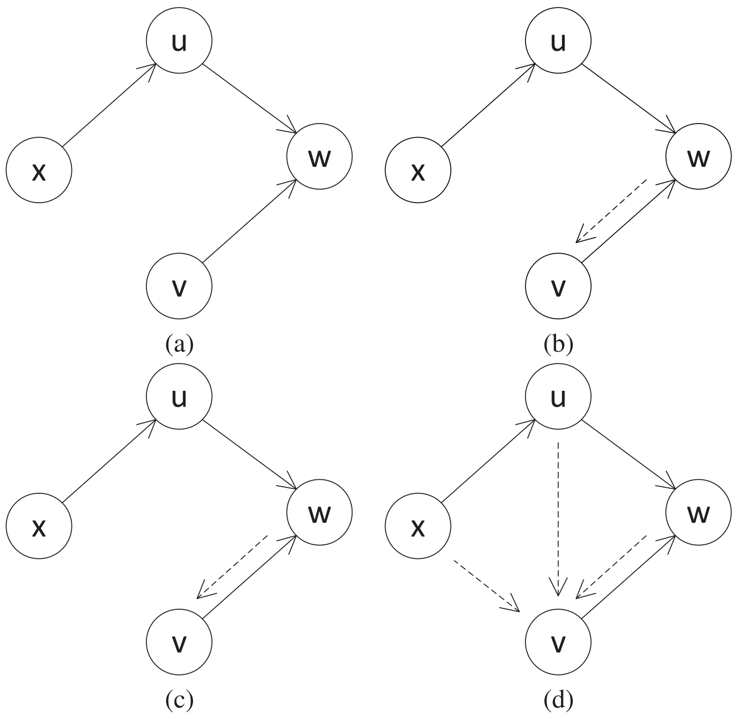


Fig. 2. Coupling may result from transposition and forwarding.

and $w \to v$ exist, edge $u \to v$ also exists by forwarding (c). Finally since both edges $x \to u$ and $u \to v$ exist, edge $x \to v$ exists by forwarding (d). This proves that coupling may result from forwarding and transposition. □

Due to this lemma, we choose to ignore cocitation and coupling, and consider the atomic trust propagation mechanisms to only contain transposition and forwarding. An immediate benefit of this simplification is that later in the model we construct, fewer parameters will be needed.

3.3. Probabilistic Trust Propagation Hypotheses

A main leap of our work from TP is that instead of considering the atomic propagation mechanisms to occur deterministically, we consider them to take effect probabilistically. This consideration accommodates the uncertainty arising from statistical irregularity, where, for example, an edge that should appear by propagation does not. Such a treatment also accounts for the possible existence of other social mechanisms, beyond trust propagation, that causes new edges to appear.

To that end, let directed graph $\mathcal{C}$ be the network containing all potential trust relationships that may possibly be formed by trust propagation. That is, the observed network $\mathcal{N}$ is a proper spanning subgraph of $\mathcal{C}$, namely that the vertex set $\mathcal{U}(\mathcal{N})$ of the network $\mathcal{N}$ is the same as the vertex set $\mathcal{U}(\mathcal{C})$ of the network $\mathcal{C}$ and the edge set $\mathcal{E}(\mathcal{N})$ of $\mathcal{N}$ is strictly contained in the edge set $\mathcal{E}(\mathcal{C})$ of $\mathcal{C}$. We will call the network $\mathcal{C}$ the *candidate network*. It is worth noting that the notion of candidate network here is meant to include the set of all edges that may *possibly* appear under trust propagation. In other words, for any pair of vertices (u, v), if (u, v) does not form an edge in the candidate network $\mathcal{C}$, then we consider it impossible for u to trust v via the propagation of trust. How to decide the candidate network is a matter of subtle technicality, which we postpone to a later section of this article.

Given the candidate network $\mathcal{C}$, the trust prediction problem can be phrased as predicting whether every edge in $\mathcal{E}(\mathcal{C})\backslash\mathcal{E}(\mathcal{N})$ will eventually appear as the social network evolves.

For every edge $i \to j$ of the candidate network $\mathcal{C}$, we introduce a $\{0, 1\}$-valued random variable $X_{i\to j}$ where $X_{i\to j} = 1$ indicates the event that the edge $i \to j$ exists or will eventually appear (whether or not it is already contained in network $\mathcal{N}$), and $X_{i\to j} = 0$ indicates the event that the edge $i \to j$ will never appear.

Now we introduce two probabilistic hypotheses that "soften" the transposition and forwarding mechanisms.

Hypothesis 1 (*Probabilistic Transposition*). For every pair of vertices i and j with both $i \to j \in \mathcal{E}(\mathcal{C})$ and $j \to i \in \mathcal{E}(\mathcal{C})$, the conditional distribution $p_{X_{i\to j}|X_{j\to i}}$ of $X_{i\to j}$ conditioned on $X_{j\to i}$ is function $t(\cdot)$ defined as follows.

$$t(x_{i\to j}, x_{j\to i}) = \begin{cases} \alpha, & x_{i\to j} = 1, x_{j\to i} = 1 \\ \alpha, & x_{i\to j} = 0, x_{j\to i} = 0 \\ 1-\alpha, & x_{i\to j} = 1, x_{j\to i} = 0 \\ 1-\alpha, & x_{i\to j} = 0, x_{j\to i} = 1 \end{cases} \tag{1}$$

for some $\alpha \in (0, 1)$.

We note that the function $t(\cdot)$ may be rewritten in the standard form of (conditional) Bernoulli distribution and the symmetry of $t(\cdot)$ in its two arguments also makes it possible to interpret the function $t(\cdot)$ reciprocally as $p_{X_{j\to i}|X_{i\to j}}$. In a nutshell, the hypothesis states that transposition occurs with probability α.

For every pair of vertices i and j, if there is a vertex k for which both $i \to k$ and $k \to j$ are in $\mathcal{E}(\mathcal{C})$, then we say that vertex k is a *bridge vertex* from i to j. We will use $\Gamma(i, j)$ to denote the set of all bridge vertices from i to j. Collectively we denote random variables $\{X_{i\to k} : k \in \Gamma(i, j)\}$ by $X_{i\to\Gamma(i,j)}$. The vector configuration taken by $X_{i\to\Gamma(i,j)}$ is then denoted by the corresponding lower-case notation $x_{i\to\Gamma(i,j)}$. Likewise, $X_{\Gamma(i,j)\to j}$ and $x_{\Gamma(i,j)\to j}$ denote respectively random variables $\{X_{k\to j} : k \in \Gamma(i, j)\}$ and their vector configuration.

For any two binary ($\{0, 1\}$-valued) vectors a and b of the same length, let $\cap(a, b)$ denote the number of positions at which both vectors a and b have value 1. With this notation, $\cap(x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j})$ is the number of bridge vertices k between vertices i and j such that both $x_{i\to k}$ and $x_{k\to j}$ equal 1.

Hypothesis 2 (*Probabilistic forwarding*). For every edge $i \to j \in \mathcal{E}(\mathcal{C})$, the conditional distribution $p_{X_{i\to j}|X_{i\to\Gamma(i,j)},X_{\Gamma(i,j)\to j}}$ of $X_{i\to j}$ conditioned on $X_{i\to\Gamma(i,j)}$ and $X_{\Gamma(i,j)\to j}$ is function $q(\cdot)$ defined as follows.

$$q\big(x_{i\to j}, x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j}\big) = (1-\Delta)^{x_{i\to j}}\,\Delta^{1-x_{i\to j}}, \tag{2}$$

where

$$\Delta = (1-\eta)(1-\beta)^{\cap(x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j})}$$

and $\eta, \beta \in (0, 1)$.

Equation (2) has an interesting physical interpretation: Imaging that every bridge vertex k from i to j potentially forms a "bridge" from i to j. More specifically, the bridge is formed if and only if $X_{i\to k} = X_{k\to j} = 1$ (namely, i trusts k and k trusts j). Given that a bridge from i to j is formed, forwarding along that bridge is assumed to occur with probability β and not to occur with probability $1 - \beta$. If more than one bridges are formed, we assume that forwarding events along these bridges are independent. Then the probability that forwarding does not occur along any of the bridges is $(1-\beta)^{\cap(x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j})}$ noting that the number of bridge vertices k between i and j such that i trusts k and k trusts j is precisely $\cap(x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j})$. Furthermore, we assume that forwarding may occur along an "alternative route", beyond these bridges, and such an event, independent of the forwarding events along the bridges, is assumed to have probability η. In such a scenario, it is easy to verify that Δ in the above equation is precisely the probability that no forwarding event occurs along any of the bridges or along the alternative route. This gives rise to the Bernoulli form of distribution $q(\cdot)$. We note that the introduction of this "alternative route," or multiplicative term $(1-\eta)$, serves two purposes: first, it serves as a regularization term which softens the effect of forwarding along the bridges; second, it accommodates additional link formation mechanisms beyond those that can be modelled as trust propagation.

4. PROBABILISTIC TRUST PROPAGATION MODEL

4.1. The Model

The probabilistic propagation hypotheses in previous section have served to quantitatively model the probabilistic propagation mechanisms, where the parameters in the hypotheses are equipped with probabilistic meanings. To arrive at a complete probabilistic model, we need to express the probabilistic relationship between the unobserved true trust relationships in candidate network $\mathcal{C}$ and the observed network $\mathcal{N}$.

For each edge $i \to j \in \mathcal{E}(\mathcal{C})$, introduce another $\{0,1\}$-valued random variable $Y_{i\to j}$, which indicates if edge $i \to j$ is contained in the observed network $\mathcal{N}$ (1 for "contained").

Hypothesis 3 (*Probabilistic Observation*). Conditioned on the trust relationships $X_{\mathcal{E}(\mathcal{C})}$ on the candidate network, observed random variables $Y_{\mathcal{E}(\mathcal{C})}$ are independent. Furthermore, for each edge $i \to j \in \mathcal{E}(\mathcal{C})$, the conditional distribution $p_{Y_{i\to j}|X_{i\to j}}$ of $Y_{i\to j}$ conditioned on $X_{i\to j}$ is the function $o(\cdot)$ defined as follows.

$$o(x_{i\to j}, y_{i\to j}) = \begin{cases} \delta, & x_{i\to j}=1, y_{i\to j}=1 \\ 1, & x_{i\to j}=0, y_{i\to j}=0 \\ 1-\delta, & x_{i\to j}=1, y_{i\to j}=0 \\ 0, & x_{i\to j}=0, y_{i\to j}=1 \end{cases} \tag{3}$$

for some value $\delta \in (0,1)$.

This hypothesis states that whether there is an observed trust relationship in $\mathcal{N}$ from user i to user j only depends on whether i trusts (or will trust) j and nothing else. This dependency is probabilistic: if user i will never trust user j, then with probability 0 the edge $i \to j$ is observed in $\mathcal{N}$; on the other hand, if user i trusts user j, then with probability δ this trust relationship is observed in $\mathcal{N}$. In essence, random variable $X_{i\to j}$ here may be understood as the *true* (eventual) existence of trust relationship from i to j, and it is hidden; what is available instead is the corresponding random variable $Y_{i\to j}$, indicating whether this trust relationship is observed in the network $\mathcal{N}$. At the first glance, it may appear too stringent to assume that no trust from i to j can be observed in $\mathcal{N}$ given i does not trust j. This assumption is in fact appropriate, since it rarely occurs that a user accidentally casts a trust on some one he does not trust. In addition, most of the social networking sites allow the users to withdraw a casted trust; even in the case that a user casts an unintended trust accidentally, he would usually withdraw the trust.

With this hypothesis, it follows that the joint distribution between the true trust relationships $X_{\mathcal{E}(\mathcal{C})}$ and the observed trust relationships $Y_{\mathcal{E}(\mathcal{C})}$ factors as

$$p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}\left(x_{\mathcal{E}(\mathcal{C})}, y_{\mathcal{E}(\mathcal{C})}\right) = p_{X_{\mathcal{E}(\mathcal{C})}}\left(x_{\mathcal{E}(\mathcal{C})}\right)\prod_{i\to j\in\mathcal{E}(\mathcal{C})} o(x_{i\to j}, y_{i\to j}). \tag{4}$$

To complete the model, we need to express the joint distribution $p_{X_{\mathcal{E}(\mathcal{C})}}$ on the candidate network $\mathcal{C}$ parametrically. For that purpose, we simply treat each conditional distribution introduced in the probabilistic propagation hypotheses as a *potential function* and model $p_{X_{\mathcal{E}(\mathcal{C})}}$ as the product of all these potential functions (up to scale), namely,

$$p_{X_{\mathcal{E}(\mathcal{C})}} \propto \prod_{i\to j\in\mathcal{E}(\mathcal{C}), i>j} t(x_{i\to j}, x_{j\to i})\prod_{i\to j\in\mathcal{E}(\mathcal{C})} q\left(x_{i\to j}, x_{i\to\Gamma(i,j)}, x_{\Gamma(i,j)\to j}\right). \tag{5}$$

In the first term of the Equation (5), the product is over all edges $i \to j \in \mathcal{E}(\mathcal{C})$ with $i > j$. We note that the condition "$i > j$" (apparently treating vertex indices as numbers) is simply to assure, noting the symmetry of function $t(\cdot)$, that if the $t(\cdot)$ function that involves any given variable $x_{i\to j}$ does not appear twice in the product.

Equations (1) through (5) completely describe a probabilistic model for true trust relationships and those observed, where expanding (4) gives rise to the fully parametrized joint distribution $p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}$ as a product of potential functions, $o(\cdot)$, $t(\cdot)$, and $q(\cdot)$. We note that although in the model the scaling constant (5) is not given,[4] it does not preclude efficient inference algorithms under the model.

It is worth noting that although $t(\cdot)$ and $q(\cdot)$ are used as the conditional distributions that characterize probabilistic transposition and forwarding respectively, when they are used as potential functions in this factorized model, the interaction of these functions in the global model makes it no longer rigorous to interpret them as those conditional distributions; the correct conditional distributions resulting from the model will in fact include the effect of the all potential functions interacting in the model. Nevertheless, the physical meanings of $t(\cdot)$ and $q(\cdot)$ stay similar as before, and we still encourage the conditional-distribution interpretation for the purpose of selecting the parameters for these functions. We wish to note, however, that in the global model it remains perfectly valid to interpret function $o(\cdot)$ as the conditional distribution $p_{Y_{i\to j}|X_{i\to j}}$ resulting from the model. In a factorized model whether a potential function can be interpreted as a conditional distribution is a matter of subtle technicality, on which we choose not to elaborate. The interested reader is referred to [Lauritzen 1996] for a careful treatment of this subject.

Finally, we note that the factorized model described by Equations (1) through (5) allows us to express the joint distribution $p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}$ using probabilistic graphical models. Markov random field [Kindermann and Snell 1980] and factor graph [Kschischang et al. 2001] models are both natural and in fact equivalent choices for this purpose. In this article, we choose the factor graph framework to represent the joint distribution $p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}$.

4.2. The Model Expressed Using Factor Graph

A factor graph [Kschischang et al. 2001] is bi-partite graph which contains two types of vertices, variable vertices and function vertices. Each variable vertex represents a variable, and each function vertex represents a function. A variable vertex is connected to a function vertex by an edge if it is an argument of the function. Globally the factor

[4]In fact it is in general intractable to compute the scaling constant, which is known as partition function. See, for instance, Al-Bashabsheh and Mao [2014].

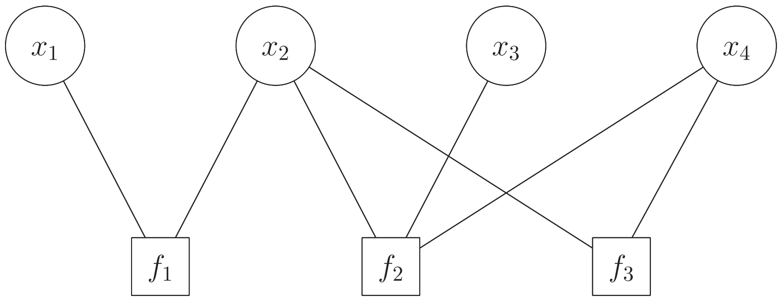


Fig. 3. The factor graph representing a multivariate function $F(x_1, x_2, x_3, x_4) = f_1(x_1, x_2) \cdot f_2(x_2, x_3, x_4) \cdot f_3(x_2, x_4)$.

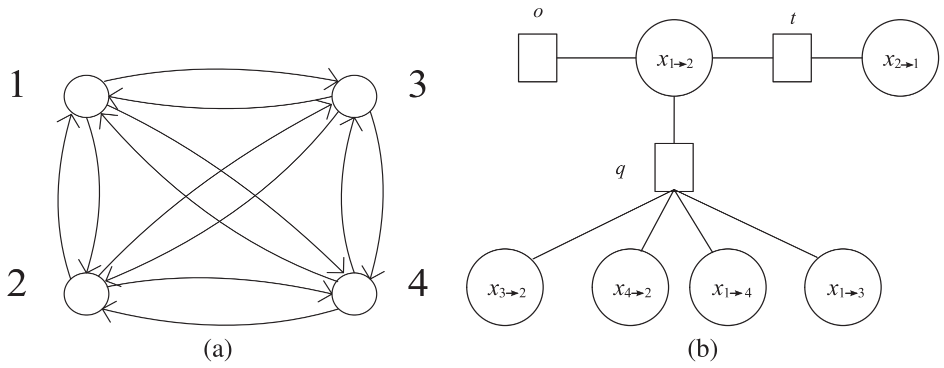


Fig. 4. Factor graph representation of probabilistic trust propagation model. (a) a toy candidate network; (b) a portion of the factor graph representation of probabilistic trust propagation model (including only variable $x_{1\to2}$ and its involving functions.)

graph represents a multivariate function that is the product of all functions represented by the function vertices. Figure 3 is an example of a factor graph.

The fact that a factor graph encodes the product of multivariate functions immediately allows us to express the joint distribution $p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}$ as a factor graph. This is illustrated in the toy example in Figure 4.

Figure 4(a) shows a toy candidate network. To create the factor graph for our probabilistic trust propagation model, we associate with each directed edge $i \to j$ a variable vertex $x_{i\to j}$ and create function vertices according to (1) to (3), and then connect the function vertices with their variable vertices. We note that for function $o(\cdot)$, we choose to have their y-variables evaluated at the observed y-configurations, where $y_{i\to j} = 1$ indicates that i is observed to trust j in $\mathcal{N}$ and $y_{i\to j} = 0$ indicates that such a trust relationship is not observed. This way, the y-variables are not required to be present in the factor graph. The entire factor graph then represents the joint distribution $p_{X_{\mathcal{E}(\mathcal{C})},Y_{\mathcal{E}(\mathcal{C})}}$ with the y-variables set to the observed configuration. We denote this partially evaluated distribution, a function of variables $x_{\mathcal{E}(\mathcal{C})}$ only, by $F(x_{\mathcal{E}(\mathcal{C})})$.

Figure 4(b) is a portion of the factor graph created for this toy candidate network, including only variable $x_{1\to2}$ and all its functions (and their variables). We choose not to show the entire factor graph in order to avoid the intersecting edges and to achieve better clarity.

5. BELIEF PROPAGATION

With the above proposed probabilistic trust propagation model, the problem of trust prediction can be formulated as a statistical inference problem.

For simplicity, we denote by $\mathcal{H}$ the set of edges in the candidate network $\mathcal{C}$ but not in the observed network $\mathcal{N}$, namely $\mathcal{H} = \mathcal{E}(\mathcal{C})\setminus\mathcal{E}(\mathcal{N})$. we will also simplify the notation of $\mathcal{E}(\mathcal{N})$ to $\bar{\mathcal{H}}$. Then the trust prediction problem on the propagation model given in the previous section can be formulated as determining, for each $i \to j \in \mathcal{H}$, the distribution of $X_{i\to j}$ given the observed trust relationships, namely, as computing $p_{X_{i\to j}|Y_{\mathcal{H}}=0,Y_{\bar{\mathcal{H}}}=1}$. Here, 0 and 1 in the subscripts denote respectively the all-zero and all-one configurations. Reading off $p_{X_{i\to j}|Y_{\mathcal{H}}=0,Y_{\bar{\mathcal{H}}}=1}(1)$ gives rise to the probability that i will eventually trust j.

It is easy to verify that $p_{X_{i\to j}|Y_{\mathcal{H}}=0,Y_{\bar{\mathcal{H}}}=1} \propto \sum_{\sim x_{i\to j}} F(x_{\mathcal{E}(\mathcal{C})})$, where the summation is over all variables of function $F(\cdot)$ except $x_{i\to j}$. When the function f is represented by a factor graph, it is well-known that simultaneous computation of $\sum_{\sim x_{i\to j}} F(x_{\mathcal{E}(\mathcal{C})})$ for every $i \to j \in \mathcal{H}$ can be carried out in parallel using the belief propagation (BP) algorithm [Kschischang et al. 2001] (also known as the sum-product algorithm).

Briefly, BP simultaneously computes many marginal functions[5] of a given multivariate function by passing "messages" along the edges of the factor graph representing the function. A message passed to or from a variable vertex is a function of that variable and a message is computed using all messages coming from all directions except that coming from the destination direction. Also computed in the belief propagation algorithm are the "summary messages" associated with the variable vertices and the summary message at each variable vertex is computed using the incoming messages from all directions. The precise message-updating rules in BP are given in Appendix. The reader is also referred to Kschischang et al. [2001] for a comprehensive tutorial.

When the factor graph does not contain any cycle, it can be shown that the summary messages in the end[6] are precisely the desired marginal functions. When the factor graph contains cycles, experimental results have shown that after some number of iterations the summary messages are good approximations of the desired marginal functions, particularly for large factor graphs. This understanding has in fact led to the a revolution in the area of error control coding, where the application of BP has allowed communication engineers to achieve the theoretical limit of digital communications [Richardson and Urbanke 2001].

In this work, we apply BP to our probabilistic trust propagation model (the factor graph representing function $F(\cdot)$). The local message-update rules are given as follows.

—The message passed from the a variable vertex to an adjacent function vertex is computed as the product of all messages incoming from all other adjacent function vertices. See Equation (17) in Appendix A.1.

—The message passed from a function vertex to an adjacent variable vertex follows from Equation (16) in Appendix A.1. On our probabilistic trust propagation model, the messages may be reduced to the following forms, depending on the type of the function vertex.

  —When the function vertex is a $t(\cdot)$ function, the message passed from the function t to its adjacent variable, say, x, is updated as

$$\begin{aligned}\mu_{t\to x}(0) &= \sum_{x'} t(0,x')\mu_{x'\to t}(x')\\ \mu_{t\to x}(1) &= \sum_{x'} t(1,x')\mu_{x'\to t}(x'),\end{aligned} \tag{6}$$

  where x' is the other variable involved in the function t (noting that t has two variables).

[5]A marginal function of function $f(\cdot)$ is the sum of function over all of its variables except one.
[6]On cycle-free graphs, BP terminates in finite number of iterations.

—When the function vertex is an $o(\cdot)$ function, the message passed from the function o to its *only* adjacent variable, again denoted by x, is simply

$$\begin{aligned}\mu_{o\to x}(0) &= t(0,x)\\ \mu_{o\to x}(1) &= t(1,x).\end{aligned} \tag{7}$$

—When the function vertex is a $q(\cdot)$ function, the message passed from the function q to its adjacent variable x is updated as follows.

If x is the variable $x_{i\to j}$ in Equation (2), we rewrite $q(\cdot)$ as $q(x,\vec{y},\vec{z})$, where $\vec{y}$ and $\vec{z}$ are respectively $x_{i\to\Gamma(i,j)}$ and $x_{\Gamma(i,j)\to j}$. Then

$$\mu_{q\to x}(0) = \sum_{\vec{y}\vec{z}} q(0,\vec{y},\vec{z}) \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i) \tag{8}$$

$$\mu_{q\to x}(1) = \sum_{\vec{y}\vec{z}} q(1,\vec{y},\vec{z}) \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i). \tag{9}$$

If x is $x_{i\to k}$ in Equation (2) for some k in $\Gamma(i,j)$, we denote $x_{k\to j}$ in Equation (2) by x'; if x is $x_{k\to j}$ in Equation (2) for some k in $\Gamma(i,j)$, we denote $x_{i\to k}$ in Equation (2) by x'. In both cases, we use $\vec{y'}$ and $\vec{z'}$ to denote respectively the vector configuration $x_{i\to\Gamma(i,j)\backslash k}$ and the vector configuration $x_{\Gamma(i,j)\backslash k\to j}$ in Equation (2). In both cases, the message passed from function q to variable x takes the following form.

$$\begin{aligned}\mu_{q\to x}(0) = \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \Bigg[& q(x_{i\to j}, <x=0,\vec{y'}>, <x',\vec{z'}>) \\ & \prod_{y'_i\in\vec{y'}} \mu_{y'_i\to q}(y'_i) \prod_{z'_i\in\vec{z'}} \mu_{z'_i\to q}(z'_i)\Bigg]\end{aligned} \tag{10}$$

$$\begin{aligned}\mu_{q\to x}(1) = \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \Bigg[& q(x_{i\to j}, <x=1,\vec{y'}>, <x',\vec{z'}>) \\ & \prod_{y'_i\in\vec{y'}} \mu_{y'_i\to q}(y'_i) \prod_{z'_i\in\vec{z'}} \mu_{z'_i\to q}(z'_i)\Bigg].\end{aligned} \tag{11}$$

For implementation purpose, these update equations can be further expanded as in Appendix A.2.

—The summary message at each variable vertex is computed as the product of all messages incoming from all adjacent function vertices. See Equation (18) in Appendix A.1.

On factor graphs with cycles, the update of message can be carried out in various orders. For our probabilistic trust propagation model, we adopt the following "flooding schedule" [Kschischang et al. 2001]).

```
BEGIN
  All o−function vertices update messages;
  While (not Converged or not reaching max number of iterations)
        All q−function vertices pass messages;
        All t−function vertices pass messages;
        All variable vertices pass messages;
  End
  Compute and output summary messages at all variable vertices
END
```

We note that in the computation of all messages, it is necessary that the messages are normalized after each update. More precisely, after each message, say $m(x)$, is computed, a scaling factor $1/\sum_x m(x)$ is multiplied to function $m(\cdot)$ so that the resulting message satisfies $\sum_x m(x) = 1$. This normalization on one hand prevents underflow and overflow and on other hand gives the messages an interpretation as a probability distribution. The summary message at variable $x_{i\to j}$ evaluated at $x_{i\to j} = 1$ may then be interpreted as the probability that i trusts j.

At this point, we have completely described the BP algorithm for trust prediction, except that we yet to explain how the candidate network $\mathcal{C}$ is constructed. As this is a matter relating to the complexity of the algorithm, we leave it in the next section.

6. COMPLEXITY CONTROL OF BP

Let C denote the maximal vertex degree (counting both in-degree and out-degree) in the candidate network $\mathcal{C}$, and let N denote the number of vertices in $\mathcal{C}$, which will be made the same as the number of vertices in the observed network $\mathcal{N}$. Let L denote the number of message-passing iterations in BP. It is easy to verify that the complexity of BP is $O(NC2^C L)$. It is then clear that the complexity is governed by the local complexity (maximal vertex degree) of the network and the network size only plays the role of a linear factor. It follows that once the local complexity is manageable, the algorithm scales well to large networks.

But real networks often contain vertex with high degrees, making 2^C large number and the algorithm unaffordable. We introduce two methods to reduce the local complexity.

6.1. Construction of Low-Degree Candidate Network

The first approach we consider is a careful construction of the candidate network $\mathcal{C}$ so that it does not contain high-degree vertices.

For that purpose, we first delete vertices in the network $\mathcal{N}$ with degree higher than a pre-determined positive number J. This can be justified from two perspectives. First, a popular star followed or trusted by many fans is unlikely to trust back his fans. Second, for a user easily trusting many people, the trust votes he casted are less likely to influence other users' trust votes. For these reasons, a vertex with high degree in fact plays rather insignificant role in the propagation of trust.

After the observed network $\mathcal{N}$ is modified so that its vertex degrees are upper bounded by J, we propose to construct a sequence of candidate networks $\mathcal{C}_i$ from $\mathcal{N}$ using the following recursive formula.

$$\mathcal{C}_1 = \mathcal{N}; \tag{12}$$

$$\mathcal{C}_{i+1} = sgn(\mathcal{C}_i \cdot \mathcal{N} + \mathcal{N}^T). \tag{13}$$

We note in the above formula, we have made an abuse of notation where the notation for a network is also used to denote the adjacency matrix of the network. The function

$sgn(\cdot)$ in the formula is the function that sets the matrix entry to 1 if it is positive and sets it to 0 otherwise. When using this recursive formula to construct candidate networks, it is necessary to set the diagonal entries of $\mathcal{N}$ to 1. It is easy to verify that this recursion (Equation (13)) essentially imposes one round of deterministic forwarding followed by one round of deterministic transposition on the previous candidate network.

It is straight-forward to check that the network $\mathcal{C}_l$ constructed this way has maximal degree upper-bounded by $\Theta(J^l)$.

The question which network $\mathcal{C}_l$ to choose as our candidate network $\mathcal{C}$ is investigated experimentally in Section 7.2.

6.2. Reduced Construction of Factor Graph

Suppose that we choose network $\mathcal{C}_l$ is chosen as the candidate network. The term J^l as the maximal degree C in the candidate network still leads to a high value of 2^C even for moderate value of J. We note that the term 2^C results from the function vertices in the factor graph connecting to C binary variables. To decrease this term to a manageable level, instead of constructing the factor graph faithfully according to probability model, for the complex function vertices, namely, function $q(\cdot)$, we reduce its set of variables to a smaller set. More specifically, instead of considering all bridge nodes from i to j, we only use a randomly selected small subset of S bridge nodes to form function $q(\cdot)$. We note that this treatment does not reduce the total number of variables in the factor graph, but simply corresponds to deleting some edges connected to the $q(\cdot)$ function nodes. The justification of this randomized factor-graph construction is that a small random set of bridge nodes are sufficient to bias the forwarding probability (to 1 or to 0) and the statistical irregularity due to randomization will actually get averaged out on large graphs.

With this approach, the maximal number of variables involved in the $q(\cdot)$ functions becomes $2S + 1$. Combining the two methods, the overall complexity of BP is brought down to $O(NLJ^l 4^S)$, where S and L are small numbers and J can be moderately large. This makes the algorithm have manageable complexity and scale well to large networks.

7. EXPERIMENTS

7.1. Datasets

We performed experiments on trust relationship data from two product review web sites, Ciao[7] and Epinions.[8] The datasets we use are those used in Tang et al. [2012a] for the purpose of analyzing multi-faceted trust relationships between social network users.[9] The Ciao dataset contains 2317 users and 2122 pairs of trust relationships. The Epinion dataset contains 4877 users and 3526 pairs of trust relationship.

Let directed graph $\mathcal{G}$ represent the trust relationships in a dataset of interest (i.e., Ciao data or Epinion data) where users with degree higher than J are removed. Here, we choose $J = 15$. We then randomly masked 25% of the edges from $\mathcal{G}$ and the resulting graph was used to simulate the observed network $\mathcal{N}$.

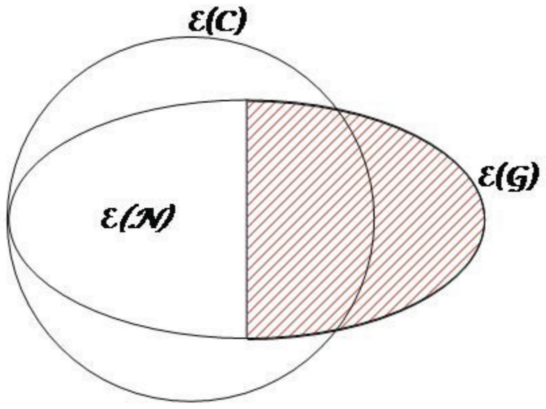
7.2. Choice of $\mathcal{C}_l$

Which value of l to choose in the construction of the candidate network $\mathcal{C}_l$ is a matter of fundamental nature and deserves careful investigation. Ideally, every edge (i, j) in

[7]www.ciao.co.uk.

[8]www.epinion.com.

[9]The datasets are publicly available at www.public.asu.edu/ jtang20/datasetcode/truststudy.htm. This work is however not compared with [Tang et al. 2012a] since additional information was used in Tang et al. [2012a].



$\mathcal{E}(\mathcal{C})$
$\mathcal{E}(\mathcal{G})$
$\mathcal{E}(\mathcal{N})$

Fig. 5. The relationship between $\mathcal{G}$, $\mathcal{N}$ and $\mathcal{C}$.

the candidate network $\mathcal{C}$ (but not in the network $\mathcal{N}$) should be such that if i trusts j then this trust relationship is due to the propagation of trust relationships in the network $\mathcal{N}$. However, the real network $\mathcal{G}$ may inherently contain trust relations that are formed for reasons beyond the propagation of trust, and such trust relationships are not reconstructable by trust propagation on $\mathcal{N}$. In addition, it is not the case that every trust relationship that should exist due to the propagation of trust has actually been established in the real network $\mathcal{G}$. This is on one hand due to that the network $\mathcal{G}$ may not have evolved to a “saturation” point where trusts have fully propagated, and on the other hand due to the fact that the trust relationship data is inherently noisy.

We use Figure 5 for further elaboration. In the figure, the ellipsoid represents the edges in $\mathcal{G}$, the shaded area represents the deleted edges from $\mathcal{G}$, and the remaining part of the ellipsoid represents the edges in $\mathcal{N}$; the circle represents the edges in $\mathcal{C}$. It is clear that the candidate network $\mathcal{C}$ should be made to include the observed network $\mathcal{N}$ as a subgraph, in order to exploit all information contained in $\mathcal{N}$.

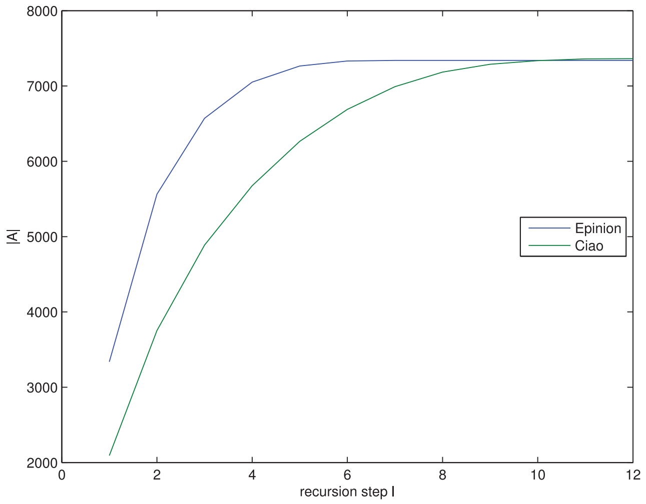
We will denote by A the set of edges that are in the circle ($\mathcal{E}(\mathcal{C})$) but outside the ellipsoid ($\mathcal{E}(\mathcal{G})$) and denote by B the set of edges in the shaded area but outside of the circle ($\mathcal{E}(\mathcal{C})$). That is,

$$A := \mathcal{E}(\mathcal{C}) \setminus \mathcal{E}(\mathcal{G}) \tag{14}$$

$$B := \mathcal{E}(\mathcal{G}) \setminus \mathcal{E}(\mathcal{C}). \tag{15}$$

Even when the candidate network $\mathcal{C}$ is chosen ideally by a genie, the size of B is the number of trust relationships that any trust propagation based algorithm is incapable of reconstructing. That is, the size $|B|$ contributes to the false negative rate. On the other hand, even for this ideal choice of $\mathcal{C}$ and for a perfect trust prediction algorithm that is based on trust propagation, due to the unsaturated and noisy nature of the network $\mathcal{G}$, there should still exist a subset A' of A, in which every edge should exist by trust propagation but not observed in $\mathcal{N}$. The size of A' will then be regarded as the number of nonexisting trust relationships that are reconstructed mistakenly. That is, the size $|A'|$ contributes to the false-positive rate.

This issue is further complicated in reality by the fact that the “ground truth” is unavailable: which trust relationships are established due to trust propagation and which trust relationships should have existed due to trust propagation are completely unknown. Nevertheless, it is expected that increasing the size of the candidate network (or the value of l for $\mathcal{C}_l$) may have the benefit of decreasing the size $|B|$, but it may also suffer from increasing the size $|A'|$ (as a consequence of increasing $|A|$). That is, the



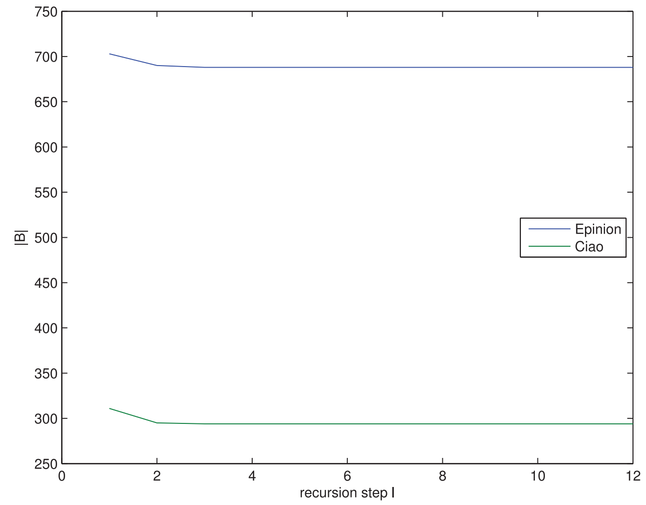


Fig. 6. The values of $|A|$ and $|B|$ as functions of l. Top: The values of $|A|$; bottom: The values of $|B|$.

value of l plays a tradeoff role between the false-positive rate and the false-negative rate.

To investigate this tradeoff and to decide the value of l for $\mathcal{C}_l$, we constructed a sequence of $\mathcal{C}_l$ and plot the values of $|A|$ and $|B|$ in Figure 6 as functions of l. From the figure, it is interesting to observe that when the value of l increases to 2, for both datasets, further increasing the value of l will only increase the value of $|A|$ without significantly decreasing the value of $|B|$. It is arguable then that increasing the value of l to higher than 2 will only increase the false-positive rate without decreasing the false-negative rate. As such, for both datasets, we fixed the value of l to 2, namely, choosing $\mathcal{C}_2$ as the candidate network $\mathcal{C}$.

7.3. Performance Metric

Due to the lack of ground truth and the fact that the real network $\mathcal{G}$ has missing edges as well as spurious edges, we take the metric of "top-K precision" as the performance

Table I. Top-K Precision of BP on Randomly Reduced Factor Graphs (Ciao Dataset, $\alpha = 0.45$, $\beta = 0.5$, $\eta = 0.1$, $\delta = 0.7$)

		K=20	K=50	K=100	K=200	K=500
$S = 1$	mean	0.3525	0.2680	0.2660	0.2555	0.2943
	variance	0.0343	0.0199	0.0110	0.0069	0.0016
$S = 2$	mean	0.4475	0.3090	0.2800	0.2685	0.2968
	variance	0.0380	0.0102	0.0000	0.0024	0.0012
$S = 3$	mean	0.5000	0.3200	0.2800	0.2700	0.2980
	variance	0.0000	0.0000	0.0000	0.0013	0.0006
$S = 4$	mean	0.5000	0.3200	0.2800	0.2700	0.2980
	variance	0.0000	0.0000	0.0000	0.0009	0.0002
$S = 5$	mean	0.5000	0.3200	0.2800	0.2700	0.2980
	variance	0.0000	0.0000	0.0000	0.0000	0.0000
$S = 6$	mean	0.5000	0.3200	0.2800	0.2700	0.2980
	variance	0.0000	0.0000	0.0000	0.0000	0.0000

Table II. Top-K Precision of BP on Randomly Reduced Factor Graphs (Epinions Dataset, $\alpha = 0.55$, $\beta = 0.5$, $\eta = 0.1$, $\delta = 0.7$)

		K=20	K=50	K=100	K=200	K=500
$S = 1$	mean	0.6475	0.3600	0.2340	0.1770	0.1662
	variance	0.0255	0.0092	0.0050	0.0025	0.0006
$S = 2$	mean	0.6525	0.3610	0.2375	0.1788	0.1661
	variance	0.0112	0.0045	0.0044	0.0022	0.0004
$S = 3$	mean	0.6500	0.3600	0.2400	0.1800	0.1660
	variance	0.0000	0.0000	0.0033	0.0017	0.0003
$S = 4$	mean	0.6500	0.3600	0.2400	0.1800	0.1660
	variance	0.0000	0.0000	0.0031	0.0015	0.0003
$S = 5$	mean	0.6500	0.3600	0.2400	0.1800	0.1660
	variance	0.0000	0.0000	0.0000	0.0000	0.0001
$S = 6$	mean	0.6500	0.3600	0.2400	0.1800	0.1660
	variance	0.0000	0.0000	0.0000	0.0000	0.0000

metric for evaluating trust prediction algorithms. We note that a similar approach is taken in Guha et al. [2004]. More specifically for our BP algorithm, for each edge $i \to j \in \mathcal{E}(\mathcal{C}) \setminus \mathcal{E}(\mathcal{N})$, which we refer to as a *query edge*, we compute the posterior probability that i trusts j, and sort all such edges according to this probability values. For various choices of integer K, we determine the number of deleted edges in $\mathcal{G}$ but not in B that are contained in the top-K query edges. Denoting this number by M, the top-K precision, denoted by P@K, is then computed as M/K. For other algorithms that we evaluate, the same metric is used where the sorting is based on the computed scores in the respective algorithms.

7.4. Validating Reduced Construction of Factor Graphs

One innovation in our BP implementation is the random reduction of the factor graph model (see Section 6.2) in order to reduce the algorithm complexity so that it can deal with large social networks. A set of experiments are performed to validate this approach.

Recall in Section 6.2, parameter S denotes the maximum size of the variable subset for the complex functions after random reduction. We let value S range from 1 to 6, and for each value of S, 10 randomly reduced factor graphs are constructed. On each reduced factor graph, the BP algorithm is run and the top-K precision is computed.

Tables I and II contain the mean and variance of the top-K-precision results for each value of S. The results apparently suggest that such randomized reduction only leads

Table III. Relative Frequencies of Transposition and Forwarding in Ciao and Epinions Datasets

	transposition	forwarding
ciao	0.4799	0.0888
epinions	0.3486	0.0905

Table IV. Top-K Precision of BP on Ciao Dataset, S=3)

α	β	η	δ	K=20	K=50	K=100	K=200	K=500
0.25	0.5	0.05	0.7	0.4000	0.3000	0.2800	0.2650	0.2960
0.25	0.5	0.05	0.8	0.4000	0.3000	0.2800	0.2650	0.2940
0.25	0.5	0.05	0.9	0.4000	0.3000	0.2800	0.2650	0.2940
0.25	0.5	0.1	0.7	0.4000	0.3000	0.2800	0.2650	0.2960
0.25	0.5	0.1	0.8	0.4000	0.3000	0.2800	0.2650	0.2960
0.25	0.5	0.1	0.9	0.3500	0.2800	0.2700	0.2600	0.2940
0.25	0.5	0.15	0.7	0.3500	0.2800	0.2700	0.2600	0.2940
0.25	0.5	0.15	0.8	0.3500	0.2800	0.2600	0.2550	0.2940
0.25	0.5	0.15	0.9	0.3500	0.2800	0.2700	0.2550	0.2960
0.45	0.5	0.05	0.7	0.5000	0.3200	0.2800	0.2700	0.2980
0.45	0.5	0.05	0.8	0.5000	0.3200	0.2800	0.2700	0.2980
0.45	0.5	0.05	0.9	0.4500	0.3000	0.2800	0.2700	0.2960
0.45	0.5	0.1	0.7	0.5000	0.3200	0.2800	0.2700	0.2980
0.45	0.5	0.1	0.8	0.4500	0.3200	0.2800	0.2700	0.2980
0.45	0.5	0.1	0.9	0.4500	0.3000	0.2800	0.2700	0.2960
0.45	0.5	0.15	0.7	0.4500	0.3200	0.2800	0.2700	0.2980
0.45	0.5	0.15	0.8	0.4500	0.3000	0.2800	0.2700	0.2960
0.45	0.5	0.15	0.9	0.3500	0.2800	0.2800	0.2650	0.2940
0.65	0.5	0.05	0.7	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.05	0.8	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.05	0.9	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.1	0.7	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.1	0.8	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.1	0.9	0.3000	0.2600	0.2500	0.2550	0.2940
0.65	0.5	0.15	0.7	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.15	0.8	0.3500	0.2600	0.2600	0.2550	0.2940
0.65	0.5	0.15	0.9	0.3000	0.2600	0.2500	0.2550	0.2940

to negligible variation in performance. For the rest of the experiments, we then choose $S = 3$.

7.5. Performance Comparison

Experiments are performed to compare the proposed BP algorithm with the four algorithms introduced earlier: TP, CN, AA, and KZ. In our experiments, the simulated networks $\mathcal{N}$ are obtained from Epinions and Ciao datasets. We choose various parameter settings for our BP algorithm, where instead of exhaustively searching the parameter space, we use the physical meanings of the parameters as a guideline. In particular, the parameters α and η are sampled respectively around the relative frequencies of transposition and forwarding in the datasets (Table III). The parameter β is set to 0.5 to be the least biased, and relatively large (i.e., close-to–1) values of δ are chosen. The performance of BP under these parameter settings is tabulated in Tables IV and V.

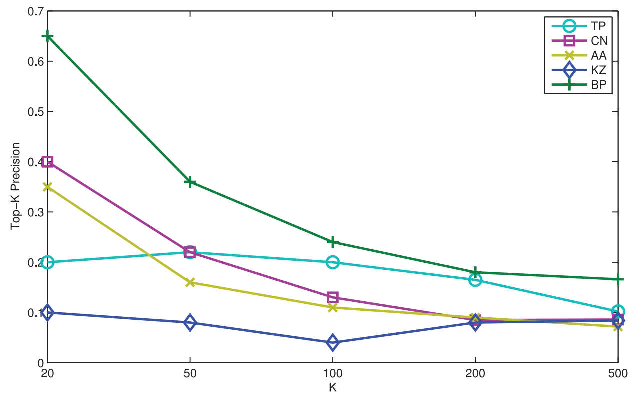
Table V. Top-K Precision of BP on Epinions Dataset, S=3

α	β	η	δ	K=20	K=50	K=100	K=200	K=500
0.15	0.5	0.05	0.7	0.3000	0.4200	0.2900	0.2050	0.1780
0.15	0.5	0.05	0.8	0.3000	0.4200	0.2900	0.2050	0.1780
0.15	0.5	0.05	0.9	0.3000	0.4200	0.2800	0.2050	0.1760
0.15	0.5	0.1	0.7	0.3000	0.4000	0.3000	0.2150	0.1780
0.15	0.5	0.1	0.8	0.3000	0.4000	0.3000	0.2150	0.1780
0.15	0.5	0.1	0.9	0.3000	0.3800	0.2900	0.2150	0.1760
0.15	0.5	0.15	0.7	0.3000	0.3600	0.2900	0.2150	0.1780
0.15	0.5	0.15	0.8	0.3000	0.3600	0.2900	0.2150	0.1780
0.15	0.5	0.15	0.9	0.3000	0.3400	0.3200	0.2200	0.1820
0.35	0.5	0.05	0.7	0.3000	0.4400	0.2800	0.2100	0.1760
0.35	0.5	0.05	0.8	0.3000	0.4000	0.2700	0.2000	0.1760
0.35	0.5	0.05	0.9	0.3000	0.4200	0.2900	0.2050	0.1780
0.35	0.5	0.1	0.7	0.3000	0.4000	0.2700	0.1950	0.1740
0.35	0.5	0.1	0.8	0.3000	0.4200	0.2900	0.2050	0.1780
0.35	0.5	0.1	0.9	0.3000	0.4200	0.2900	0.2050	0.1780
0.35	0.5	0.15	0.7	0.3000	0.4200	0.2800	0.2000	0.1760
0.35	0.5	0.15	0.8	0.2500	0.4200	0.2900	0.2050	0.1780
0.35	0.5	0.15	0.9	0.3000	0.4200	0.2800	0.2050	0.1760
0.55	0.5	0.05	0.7	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.05	0.8	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.05	0.9	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.1	0.7	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.1	0.8	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.1	0.9	0.6000	0.3600	0.2300	0.1750	0.1660
0.55	0.5	0.15	0.7	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.15	0.8	0.6500	0.3600	0.2400	0.1800	0.1660
0.55	0.5	0.15	0.9	0.6500	0.3400	0.2300	0.1750	0.1640

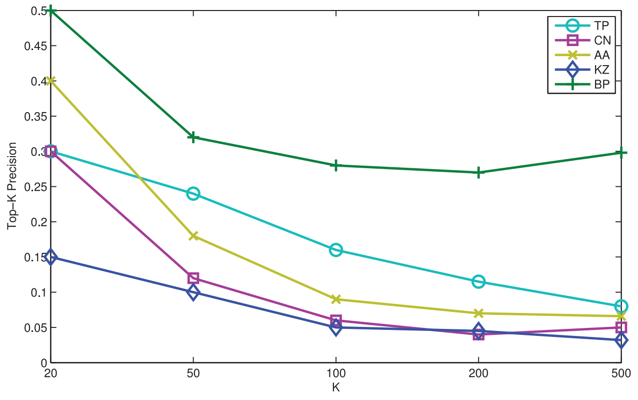
It appears that BP achieves the best top-K precision when the parameters are set to $\{\alpha = 0.45, \beta = 0.5, \eta = 0.1, \delta = 0.7\}$ for Epinions and $\{\alpha = 0.55, \beta = 0.5, \eta = 0.1, \delta = 0.7\}$ for Ciao. It is certainly possible to fine tune these parameters and further improve the performance of BP. Nonetheless we choose to use these parameter settings for BP in subsequent experiments.

For each of the four algorithms under comparison, we search their parameter space for the settings that give the best top-K precision. For TP [Guha et al. 2004], we apply four rounds of the propagation. The comparison of BP with these algorithms are conducted for various top-K precision metrics (with varying values of K). It can be observed that BP consistently outperforms all other algorithms for both Epinions and Ciao datasets and for various values of K; the TP algorithm performs as the second best algorithm, as one has expected. The other three algorithms CN, AA and KZ perform similarly, with KZ at the lowest level.The performance gap between BP and TP appears to be more significant than the gap between TP and other algorithms. This demonstrates the effectiveness of the probabilistic principle exploited in BP. (see Figure 7.)

Also observed in the plots is the general decreasing trend of top-K precision with increasing K (except for BP with $K = 500$, which is perhaps due to statistical irregularity). This is expected since relative to the total number of available user pairs, trust relationships only exist sparsely; that is, it is more likely for a user not to trust another user than to trust him. As such, when K increases, trust prediction algorithms will



(a) Epinions



(b) Ciao

Fig. 7. Top-K precision of BP, TP, CN, AA and KZ for various values of K.

tend to have its false-positive rate increased, thereby reducing the top-K precision. In the limit when K takes the value equal to the total number of query edges, it is easy to see that *every* trust prediction algorithm (beyond the ones considered in this article) will behave exactly the same, namely, giving rise to an identical top-K precision score. This inherent property of top-K precision suggests that letting K take very large value will in fact make this metric meaningless.

In summary, under the metric of top-K precision, we have shown that BP performs better than other existing algorithms. Although the lack of ground truth has made it difficult for selecting a perfect value of K when using top-K precision as the metric, our experimental results indicate that for a wide range of K, BP outperforms the other algorithms by a good margin.

8. CONCLUDING REMARKS

Predicting trust relationships in a social networks is an important task for the network providers. Although link prediction algorithms previously established for general networks may be adapted to such applications, the recent notion of trust propagation [Guha et al. 2004] has been conceptualized as a kinetics for the formation of trust relationships and has led to an effective algorithm for trust prediction.

This article builds upon the concept of trust propagation and introduces a probabilistic trust propagation model. We first show theoretically that the original four trust propagation postulates can be reduced to two only without sacrificing their expressive power. Probabilistic hypotheses are then introduced based on the reduced postulates in order to accommodate the inherent noise and uncertainty in the dataset and additional trust-formation mechanisms beyond trust propagation. These hypotheses are then complied into a probabilistic trust propagation model which is readily represented by a factor graph [Kschischang et al. 2001]. In this framework, the trust prediction problem can be formulated as a statistical inference problem, for which we derived the belief propagation algorithm as a solver. Using datasets from Epinions and Ciao, this algorithm, we show experimentally that BP outperforms the existing link prediction algorithms and the trust propagation algorithm.

We note that the similarity between this work and that of Guha et al. [2004] is the exploitation of "trust propagation" in the prediction of trust relationships. This work however distinguishes itself from Guha et al. [2004] in several aspects. First, the notion of trust propagation in Guha et al. [2004] is a deterministic notion whereas the notion of trust propagation in this work is a probabilistic one. Secondly, in this work, we formulate the trust prediction problem as an statistical inference problem whereas in Guha et al. [2004] the problem is not as rigorously formulated in mathematical terms. Thirdly, the BP algorithm derived in this article is a well principled algorithm whereas the algorithm of [Guha et al. 2004] is largely heuristic in nature.

We believe that this work is the first to formulate the trust prediction problem in a probabilistic framework as a probabilistic inference problem and the first to use a "declarative" approach to solving the problem (as opposed to the previous approaches, all of which are "procedural" in nature). The performance advantage of our approach is attributed to the principled inference framework on one hand, and to the effectiveness of the BP algorithm on the other.

An additional modest contribution of this work is a novel implementation of the BP algorithm in trust belief propagation where the factor graph model is randomly reduced for controlling the algorithm complexity. This approach, validated experimentally, allows us to deal with large social networks efficiently.

As usual, this work is by no means optimal and one expects that further improvement of the proposed algorithm may be possible. For example, in this work, the model parameters, namely, α, β, η, δ, are selected offline by considering the physical meanings of these parameters. It is also possible to develop an EM algorithm to jointly estimate these parameters while inferring the latent trust relationships at the same time. In that case, the present BP algorithm will become one step (the E step) of the EM algorithm. Such a joint estimation approach is expected to be more effective when dealing with large datasets. For datasets of moderate size, such as the ones studied in this article, the advantage of the EM algorithm may be compromised by the cost entailed by the possibility of over-fitting.

Regardless of the potential directions along which this work may be extended, at this point it appears to us that the issues of testing and performance evaluation in this line of research shall not be overlooked. In this work, we point out that the available trust relationship data contains those arising from trust propagation as well as those emerging for other reasons. Furthermore, some trust relationships (that should arise from trust propagation) are likely missing from the data. The intertwining of these two factors results is further complicated by the problem of "missing ground truth". Although this work has carefully examined this aspect, better methods for algorithm testing and additional metrics for algorithm evaluation are still desirable. Potential routes to tackle this issue may involve a novel framework to formulate the problem, along which our investigation is underway.

APPENDIXES

A.1. Message Update Rules of Belief Propagation

In belief propagation, a message passed from a vertex u to a vertex v depends on the messages passed to u from all neighbours of u except the one passed from v. Below we assume that f is a function vertex and u is a variable vertex, and that f and u are neighbours of each other. The notations $N(f)$ and $N(u)$ denote respectively the set of all neighbours of f and the set of all neighbours of u respectively. (Noting that a neighbour of variable vertex is necessarily a function vertex and vice versa).

The message $\mu_{f\to u}$ passed from f to u and the message $\mu_{u\to f}$ passed from u to f are given as follows.

$$\mu_{f\to u}(x_u) := \sum_{x_{N(f)\setminus\{u\}}} f\left(x_{N(f)}\right) \cdot \prod_{u'\in N(f)\setminus\{u\}} \mu_{u'\to f}(x_{u'}) \tag{16}$$

$$\mu_{u\to f}(x_u) := \prod_{f'\in N(u)\setminus\{f\}} \mu_{f'\to u}(x_u). \tag{17}$$

In addition, a "summary message" is computed at each variable vertex u as

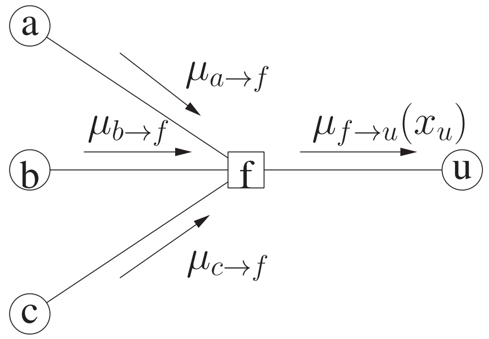
$$\mu_u(x_u) := \prod_{f\in N(u)} \mu_{f\to u}(x_u) \tag{18}$$

The reader is referred to Figure 8 for an illustration of the message-passing procedures. The passing of messages may arranged in various orders, known as "schedules". Every schedule includes three phases: initialization of messages, propagation of messages, and termination of message passing. It is well known that if the factor graph representing $F(x_V)$ contains no cycle, then with certain default initialization rule, the messages computed necessarily converge after some number of iterations, and the resulting summary message for every $v \in V$ is precisely the marginal function $\sum_{x_{V\setminus\{v\}}} F(x_V)$. In the case when the factor graph representing F contains cycles, heuristic schedules are often applied, and the resulting summary messages upon termination are not necessarily equal to the marginal functions. Experimental results have shown however that even in these cases, the summary messages are good approximations of those marginal functions, particularly when the factor graph is large (see, for example, Richardson and Urbanke [2001]).

A.2. The Derivation of Message from a Function Vertex *q* Passed to a Variable Vertex *x*

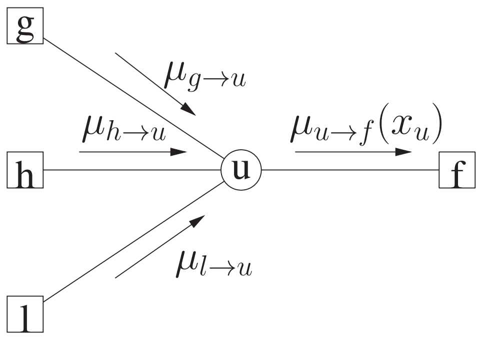
We rewrite q as $q(x, \vec{y}, \vec{z}) = (1 - \Delta(\vec{y}, \vec{z}))^x(\Delta(\vec{y}, \vec{z}))^{1-x}$, denote $1 - \eta = \bar{\eta}$, and $1 - \beta = \bar{\beta}$, suppose vectors $\vec{y}$ and $\vec{z}$ have length L. If $x = x_{i\to j}$, then

$$\begin{aligned}
\mu_{q\to x}(0) &= \sum_{\vec{y}\vec{z}} q(0, \vec{y}, \vec{z}) \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i) \\
&= \sum_{\vec{y}\vec{z}} \Delta(\vec{y}, \vec{z}) \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i) \\
&= \sum_{\vec{y}\vec{z}} \bar{\eta}\bar{\beta}^{\vec{y}\vec{z}} \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i) \\
&= \sum_{\ell=0}^{L} \sum_{(\vec{y},\vec{z}):\cap(\vec{y},\vec{z})=\ell} \bar{\eta}\bar{\beta}^{\ell} \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i).
\end{aligned} \tag{19}$$



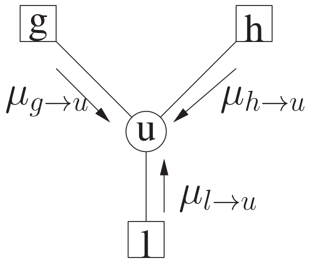
$$\mu_{f\to u}(x_u) = \sum_{x_a,x_b,x_c} f(x_a,x_b,x_c,x_u)\mu_{a\to f}(x_a)\mu_{b\to f}(x_b)\mu_{c\to f}(x_c)$$

(a) Update of a message passed to a variable vertex



$$\mu_{f\to u}(x_u) = \mu_{g\to u}(x_u)\cdot\mu_{h\to u}(x_u)\cdot\mu_{l\to u}(x_u)$$

(b) Update of a message passed to a function vertex



$$\mu_u(x_u) = \mu_{g\to u}(x_u)\cdot\mu_{h\to u}(x_u)\cdot\mu_{l\to u}(x_u)$$

(c) Update of a summary message

Fig. 8. Illustration of message update rules in belief propagation.

Let

$$\triangle(\ell) := \sum_{(\vec{y},\vec{z}):\cap(\vec{y},\vec{z})=\ell} \bar{\eta}\bar{\beta}^{\ell} \prod_{y_i\in\vec{y}} \mu_{y_i\to q}(y_i) \prod_{z_i\in\vec{z}} \mu_{z_i\to q}(z_i). \tag{20}$$

Denote by A_ℓ the set of the subsets of $\{1,2,\ldots,L\}$ in length ℓ, then, $\triangle(\ell) = \bar{\eta}\bar{\beta}^{\ell}\sum_{a\in A_\ell}(B+C+D)$, where

$$B = \prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0) \tag{21}$$

$$C = \prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\left[\sum_{s\subset\bar{a}}\prod_{j\in a\cup s}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}\cup s}\mu_{z_{j'}\to q}(0)\right] \tag{22}$$

$$D = \left[\sum_{s\subset\bar{a}}\prod_{i\in a\cup s}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}\cup s}\mu_{y_{i'}\to q}(0)\right]\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0). \tag{23}$$

Then

$$\begin{aligned}\mu_{q\to x}(0) = &\sum_{\ell=0}^{L}(1-\bar{\eta}\bar{\beta}^{\ell})\sum_{a\in A_\ell}\left(\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0)\right.\\ &+\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\left[\sum_{s\subset\bar{a}}\prod_{j\in a\cup s}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}\cup s}\mu_{z_{j'}\to q}(0)\right]\\ &\left.+\left[\sum_{s\subset\bar{a}}\prod_{i\in a\cup s}\mu_{y_i\to q}(1)\prod_{i'\in(\bar{a}\cup s)}\mu_{y_{i'}\to q}(0)\right]\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0)\right).\end{aligned} \tag{24}$$

Similarly,

$$\begin{aligned}\mu_{q\to x}(1) &= \sum_{\vec{y}\vec{z}}q(1,\vec{y},\vec{z})\prod_{y_i\in\vec{y}}\mu_{y_i\to q}(y_i)\prod_{z_i\in\vec{z}}\mu_{z_i\to q}(z_i)\\ &= \sum_{\vec{y}\vec{z}}(1-\Delta(\vec{y},\vec{z}))\prod_{y_i\in\vec{y}}\mu_{y_i\to q}(y_i)\prod_{z_i\in\vec{z}}\mu_{z_i\to q}(z_i)\\ &= \sum_{\vec{y}\vec{z}}(1-\bar{\eta}\bar{\beta}^{\vec{y}\vec{z}})\prod_{y_i\in\vec{y}}\mu_{y_i\to q}(y_i)\prod_{z_i\in\vec{z}}\mu_{z_i\to q}(z_i)\\ &= \sum_{\ell=0}^{L}(1-\bar{\eta}\bar{\beta}^{\ell})\sum_{a\in A_\ell}\left(\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0)\right.\\ &\quad+\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar{a}}\mu_{y_{i'}\to q}(0)\left[\sum_{s\subset\bar{a}}\prod_{j\in a\cup s}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}\cup s}\mu_{z_{j'}\to q}(0)\right]\\ &\quad\left.+\left[\sum_{s\subset\bar{a}}\prod_{i\in a\cup s}\mu_{y_i\to q}(1)\prod_{i'\in(\bar{a}\cup s)}\mu_{y_{i'}\to q}(0)\right]\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar{a}}\mu_{z_{j'}\to q}(0)\right).\end{aligned} \tag{25}$$

If $x = x_{i\to k}$, $k \neq j$, we denote $x_{k\to j}$ by x'. If $x = x_{k\to j}$, $k \neq i$, we denote $x_{i\to k}$ by x'. $\vec{y}'$ and $\vec{z}'$ denote the vector configuration $x_{i\to\Gamma(i,j)\setminus k}$ and the vector configuration $x_{\Gamma(i,j)\setminus k\to j}$ respectively.

We rewrite q as $q(x_{i\to j}, < x, \vec{y'} >, < x', \vec{z'} >)$ and denote $1-\eta = \bar{\eta}$, and $1-\beta = \bar{\beta}$, then,

$$\begin{aligned}
\mu_{q\to x}(0) &= \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \Bigg[q\big(x_{i\to j}, < x=0, \vec{y'} >, < x', \vec{z'} >\big) \\
&\quad \prod_{y'_i\in\vec{y'}} \mu_{y'_i\to q}\big(y'_i\big) \prod_{z'_i\in\vec{z'}} \mu_{z'_i\to q}\big(z'_i\big) \Bigg] \\
&= \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \Delta(\vec{y'}, \vec{z'}) \prod_{y'_i\in\vec{y'}} \mu_{y'_i\to q}\big(y'_i\big) \prod_{z'_i\in\vec{z'}} \mu_{z'_i\to q}\big(z'_i\big) \\
&= \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \Bigg[\sum_{\ell=0}^{L} (1-\bar{\eta}\bar{\beta}^{\ell}) \\
&\quad \sum_{a\in A_\ell} \Bigg(\prod_{i\in a} \mu_{y_i\to q}(1) \prod_{i'\in\bar{a}} \mu_{y_{i'}\to q}(0) \prod_{j\in a} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}} \mu_{z_{j'}\to q}(0) \\
&\quad + \prod_{i\in a} \mu_{y_i\to q}(1) \prod_{i'\in\bar{a}} \mu_{y_{i'}\to q}(0) \Bigg[\sum_{s\subset\bar{a}} \prod_{j\in a\cup s} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}\cup s} \mu_{z_{j'}\to q}(0) \Bigg] \\
&\quad + \Bigg[\sum_{s\subset\bar{a}} \prod_{i\in a\cup s} \mu_{y_i\to q}(1) \prod_{i'\in(\bar{a}\cup s)} \mu_{y_{i'}\to q}(0) \Bigg] \prod_{j\in a} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}} \mu_{z_{j'}\to q}(0) \Bigg) \Bigg], \qquad (26)
\end{aligned}$$

and

$$\begin{aligned}
\mu_{q\to x}(1) &= \sum_{x_{i\to j}}\sum_{x'} \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(x') \sum_{\vec{y'}\vec{z'}} \\
&\quad q\big(x_{i\to j}, < x=1, \vec{y'} >, < x', \vec{z'} >\big) \prod_{y'_i\in\vec{y'}} \mu_{y'_i\to q}\big(y'_i\big) \prod_{z'_i\in\vec{z'}} \mu_{z'_i\to q}\big(z'_i\big) \\
&= \sum_{x_{i\to j}} \Bigg[\mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(0) \sum_{\vec{y'}\vec{z'}} \Bigg[\sum_{\ell=0}^{L} (1-\bar{\eta}\bar{\beta}^{\ell}) \\
&\quad \sum_{a\in A_\ell} \Bigg(\prod_{i\in a} \mu_{y_i\to q}(1) \prod_{i'\in\bar{a}} \mu_{y_{i'}\to q}(0) \prod_{j\in a} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}} \mu_{z_{j'}\to q}(0) \\
&\quad + \prod_{i\in a} \mu_{y_i\to q}(1) \prod_{i'\in\bar{a}} \mu_{y_{i'}\to q}(0) \Bigg[\sum_{s\subset\bar{a}} \prod_{j\in a\cup s} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}\cup s} \mu_{z_{j'}\to q}(0) \Bigg] \\
&\quad + \Bigg[\sum_{s\subset\bar{a}} \prod_{i\in a\cup s} \mu_{y_i\to q}(1) \prod_{i'\in(\bar{a}\cup s)} \mu_{y_{i'}\to q}(0) \Bigg] \prod_{j\in a} \mu_{z_j\to q}(1) \prod_{j'\in\bar{a}} \mu_{z_{j'}\to q}(0) \Bigg) \Bigg] \\
&\quad + \mu_{x_{i\to j}\to q}(x)\mu_{x'\to q}(1) \sum_{\vec{y'}\vec{z'}} \Bigg[\sum_{\ell=0}^{L} (1-\bar{\eta}\bar{\beta}^{\ell+1})
\end{aligned}$$

$$\sum_{a\in A_\ell}\left(\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar a}\mu_{y_{i'}\to q}(0)\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar a}\mu_{z_{j'}\to q}(0)\right.$$
$$+\prod_{i\in a}\mu_{y_i\to q}(1)\prod_{i'\in\bar a}\mu_{y_{i'}\to q}(0)\left[\sum_{s\subset\bar a}\prod_{j\in a\cup s}\mu_{z_j\to q}(1)\prod_{j'\in\bar a\cup s}\mu_{z_{j'}\to q}(0)\right]$$
$$\left.+\left[\sum_{s\subset\bar a}\prod_{i\in a\cup s}\mu_{y_i\to q}(1)\prod_{i'\in(\bar a\cup s)}\mu_{y_{i'}\to q}(0)\right]\prod_{j\in a}\mu_{z_j\to q}(1)\prod_{j'\in\bar a}\mu_{z_{j'}\to q}(0)\right)\Bigg]\Bigg] \quad (27)$$

REFERENCES

Karl Aberer and Zoran Despotovic. 2001. Managing trust in a peer-2-peer information system. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM '01)*. ACM, New York, 310–317.

Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social Networks* 25, 211–230.

Ali Al-Bashabsheh and Yongyi Mao. 2014. On stochastic estimation of partition function. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT'14)*. 1504–1508.

Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM'11)*. ACM, New York, 635–644.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.* 30, 1–7, 107–117. http://dx.doi.org/10.1016/S0169-7552(98)00110-X

Jennifer Golbeck. 2009. Trust and nuanced profile similarity in online social networks. *ACM Trans. Web* 3, 4, Article 12.

Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of trust and distrust. In *Proceedings of the 19th International Conference on World Wide Web*. 403–412.

Roger A. Horn and Charles R. Johnson. 1990. *Matrix Analysis*. Cambridge University Press.

Mohsen Jamali and Martin Ester. 2009. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, New York, 397–406.

Glen Jeh and Jennifer Widom. 2002. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge discovery and Data Mining (KDD'02)*. ACM, New York, 538–543.

Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (March 1953), 39–43.

Ross Kindermann and J. Laurie Snell. 1980. *Markov Random Fields and Their Applications*. Contemporary Mathematics, Vol. 1.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* 47, 2, 498–519.

Steffen L. Lauritzen. 1996. *Graphical Models*. Oxford Statistical Science Series.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. ACM, New York, 641–650.

M. Levandowsky and D. Winter. 1971. Distance between sets. *Nature* 234, 5, 34.

David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*. ACM, New York, 556–559.

David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* 58, 7, 1019–1031.

Guanfeng Liu, Yan Wang, and Mehmet A. Orgun. 2010. Optimal social trust path selection in complex social networks. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI'10)*.

Guanfeng Liu, Yan Wang, and Mehmet A. Orgun. 2011. Trust transitivity in complex social networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*.

Guanfeng Liu, Yan Wang, and Duncan S. Wong. 2012. Multiple QoT constrained social trust path selection in complex social networks. In *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*. 624–631.

Hao Ma, Irwin King, and Michael R. Lyu. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development In Information Retrieval (SIGIR'09)*. ACM, New York, 203–210.

Paolo Massa and Paolo Avesani. 2007. Trust-aware recommender systems. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'07)*. ACM, New York, 17–24.

Mark E. J. Newman. 2001. Clustering and preferential attachment in growing networks. *Phys. Rev. E* 64.

Thomas J. Richardson and Rüdiger L. Urbanke. 2001. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory* 47, 2, 599–618.

Jiliang Tang, Huiji Gao, and Huan Liu. 2012a. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM'12)*. 93–102.

Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. 2012b. eTrust: understanding trust evolution in an online world. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. 253–261.

Rongjing Xiang, Jennifer Neville, and Monica Rogati. 2010. Modeling relationship strength in online social networks. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. ACM, New York, 981–990.

Cai-Nicolas Ziegler and Jennifer Golbeck. 2007. Investigating interactions of trust and interest similarity. *Decision Support Syst.* 43, 2, 460–475.

Cai-Nicolas Ziegler and Georg Lausen. 2004. Spreading activation models for trust propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*. IEEE, 83–97.

Received September 2013; revised February 2014, April 2014; accepted May 2014