Metadata of the chapter that will be visualized online

Book Title	Encyclopedia of Social Network Analysis and Mining	
Book Copyright Year	2014	
Copyright Holder	Springer Science+Business Media New York	
Title	Subgraph Extraction for Trust Inference in Social Networks	

Author	Degree	
	Given Name	**Yuan**
	Particle	
	Family Name	**Yao**
	Suffix	
	Phone	
	Fax	
	Email	yyao@smail.nju.edu.cn
Affiliation	Division	State Key Laboratory for Novel Software Technology
	Organization	Nanjing University
	Street	163 Xianlin Avenue
	Postcode	210046
	City	Nanjing
	State	Jiangsu
	Country	China
Author	Degree	
	Given Name	**Hanghang**
	Particle	
	Family Name	**Tong**
	Suffix	
	Phone	
	Fax	
	Email	tong@cs.ccny.cuny.edu
Affiliation	Division	
	Organization	CUNY city college
	City	New York
	State	NY
	Country	USA

Author	Degree	
	Given Name	**Feng**
	Particle	
	Family Name	**Xu**
	Suffix	
	Phone	
	Fax	
	Email	xf@nju.edu.cn
Affiliation	Division	State Key Laboratory for Novel Software Technology
	Organization	Nanjing University
	Street	163 Xianlin Avenue
	Postcode	210046
	City	Nanjing
	State	Jiangsu
	Country	China
Author	Degree	
	Given Name	**Jian**
	Particle	
	Family Name	**Lu**
	Suffix	
	Phone	
	Fax	
	Email	lj@nju.edu.cn
Affiliation	Division	State Key Laboratory for Novel Software Technology
	Organization	Nanjing University
	Street	163 Xianlin Avenue
	Postcode	210046
	City	Nanjing
	State	Jiangsu
	Country	China

S

Subgraph Extraction for Trust Inference in Social Networks

Yuan Yao[1], Hanghang Tong[2], Feng Xu[1], and Jian Lu[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China
[2]CUNY city college, New York, NY, USA

Synonyms

Interaction network; Subgraph discovery; Trust evaluation; Trust network; Trust prediction

Glossary

Social Network A graph in which the nodes represent the participants in the network and the edges represent relationships

Trust-Based Social Network A directed weighted graph in which the nodes represent the participants in the network, the edges represent trust relationships and the weight on each edge indicates the local trust value derived from the historical interactions

Trust Inference A mechanism to build new trust relationships based on existing ones

Subgraph A subgraph of graph G is a graph whose node set is a subset of that of G, and whose edge set is a subset of that of G restricted to the node subset

Subgraph Extraction Discovery of a subgraph from a whole graph

Definition

Trust-based social networks might contain a large amount of redundant information, making existing trust inference suffer from the scalability and usability issues. Therefore, it is natural to apply subgraph extraction as an intermediate step to speed up as well as to interpret the trust inference process.

Introduction

Trust inference, which aims to infer a trustworthiness score from the trustor to the trustee in the underlying social network, is an essential task in many real-world applications including e-commerce (Xiong and Liu 2004), peer-to-peer networks (Kamvar et al. 2003), and mobile ad hoc networks (Buchegger and Le Boudec 2004).

To date, many trust inference algorithms have been proposed, which can be categorized into two main classes (see the next section for a review): (a) path-based inference (Mui et al. 2002; Wang and Singh 2006; Hang et al. 2009; Wang and Wu 2011) and (b) component-based inference (Guha et al. 2004; Massa and Avesani 2005; Ziegler and Lausen 2005; Zhou and Hwang 2007).

R. Alhajj, J. Rokne (eds.), *Encyclopedia of Social Network Analysis and Mining*, DOI 10.1007/978-1-4614-6170-8,

Despite their own success, most of the existing inference algorithms have two limitations. The first challenge lies in *scalability* – many existing algorithms become very time-consuming or even computationally infeasible for the graphs with more than thousands of nodes. Additionally, some algorithms assume the existence of a subgraph while how to construct such a subgraph remains an open issue (Wang and Wu 2011). The second challenge is the *usability* of the inference results. Most, if not all, of the existing inference algorithms output an abstract numerical trustworthiness score. This gives a quantitative measure of *to what extent* the trustor should trust the trustee but gives few cues on *how* the trustworthiness score is inferred. This usability/interpretation issue becomes more evident when the size of the underlying graph increases, since we cannot even display the entire graph to the end users (see Fig. 9 for an example).

In this article, we propose subgraph extraction to address these challenges. The core of our subgraph extraction consists of two stages: *path selection* and *component induction*. In the first (path selection) stage, we extract a few, important paths from the trustor to the trustee. In the second (component induction) stage, we propose a novel evolutionary algorithm to generate a small subgraph based on the extracted paths. The outputs of these two stages are then used as an intermediate step to speed up the path-based inference and component-based inference algorithms, respectively. Our experimental evaluations on real graphs show that the proposed method can significantly accelerate existing trust inference algorithms (up to 2,400× speedup) while maintaining high accuracy (P-error is less than 0.04). In addition, the extracted subgraph provides an intuitive way to interpret the resulting trustworthiness score by presenting a concise summarization on the relationship from the trustor to the trustee. To the best of our knowledge, we are the first to propose subgraph extraction for trust inference. We believe that our work can improve most of the existing trust inference algorithms by (1) scaling up as well as (2) delivering more usable (i.e., interpretation-friendly) inference results to the end users.

Historical Background

We review the historical background in this section, which can be categorized into two parts: trust inference algorithms and subgraph extraction.

Trust Inference

We categorize existing trust inference algorithms into two main classes: path-based trust inference and component-based trust inference.

In the first class of path-based inference, trust is propagated along a path from the trustor to the trustee, and the propagated trust from multiple paths can be combined to form a final trustworthiness score. For example, Wang and Singh (2006, 2007) as well as Hang et al. (2009) propose operators to concatenate trust along a path and aggregate trust from multiple paths. Liu et al. (2010) argue that not only trust values but social relationships and recommendation role are important for trust inference. However, these algorithms are only suitable for small networks due to their complexity. Some other path-based trust inference algorithms, such as Mui et al. (2002) and Wang and Wu (2011), assume the existence of an extracted subgraph while how to construct such a subgraph remains an open issue (Wang and Wu 2011).

In the second class of component-based inference, EigenTrust Kamvar et al. (2003) tries to compute an objective trustworthiness score for each node in the graph. In contrast to EigenTrust, our main focus is to provide support for subjective trust metrics where different trustors can form different opinions on the same trustee. In contrast to path-based trust inference algorithms, there is no explicit concept of paths in component-based trust inference. Instead, existing subjective trust algorithms, including Guha et al. (2004), Massa and Avesani (2005), Ziegler and Lausen (2005), and Nordheimer et al. (2010), take the initial graph as input and treat trust as random walks on a Markov chain or on a graph (Richardson et al. 2003). For example, in MoleTrust (Massa and Avesani 2005) and Appleseed (Ziegler and Lausen 2005), trust propagates along the edges according to the trust values on the edges. Our subgraph extraction method not only can speed up many of these algorithms but also can provide interpretive result which is not considered by the existing algorithms.

Overall, our subgraph extraction is motivated to address the two common challenges (i.e., scalability and usability) shared by most of these existing trust inference algorithms.

Subgraph Extraction

Several end-to-end subgraph extraction algorithms are developed to solve different problems.

In the field of graph mining, Faloutsos et al. (2004) refer to the idea of electrical current where trust relationships are modeled as resistors and try to find a connection subgraph that maximizes the current flowing from source to target. Later, Tong et al. (2007) generalize the connection subgraph to directed graphs and use the subgraph to compute proximities between nodes. Similar to Tong et al., Koren et al. (2006) also try to induce a subgraph for proximity computation. In addition, Koren et al. search the k-shortest paths to provide a basis for measuring the proximity.

Recently, several algorithms are proposed for reliable subgraph extraction. Among them, Monte Carlo pruning (Hintsanen and Toivonen 2008) measures the relevance of each edge by Monte Carlo simulations and tends to remove the edge of lowest relevance one by one. The most related work is perhaps the randomized Path Covering algorithm (Hintsanen et al. 2010) which also consists of two stages of path sampling and subgraph construction. However, both Monte Carlo pruning and Path Covering tend to find a subgraph with highest probability to be connected, while we aim to find a subgraph to address the scalability and usability issues in trust inference.

The Proposed Subgraph Extraction Method

In this section, we first formalize the subgraph extraction problem for trust inference in social networks and then introduce our proposed solution which consists of two stages: path selection and component induction.

Problem Definition

Following the standard notations in the existing trust inference algorithms, we model the trust relationships in social networks as a weighted directed graph (Barbian 2011; Yao et al. 2011). The nodes of the graph represent the participants in the network, and the weight on each edge indicates the local trust value derived from the historical interactions.

We then categorize the existing trust inference algorithms into two major classes: *path-based trust inference* and *component-based trust inference*.

Definition 1 Path-Based Trust Inference
Path-based trust inference includes the approaches, which are started by the trustor, to evaluating the trustworthiness of the trustee, through a set of paths from the trustor to the trustee in the network.

Definition 2 Component-Based Trust Inference
Component-based trust inference includes the approaches, which are started by the trustor, to evaluating the trustworthiness of the trustee, through a connected component from the trustor to the trustee in the network.

Both classes belong to the subjective trust metrics (Ziegler and Lausen 2005), where different trustors can form different opinions on the same trustee. Accordingly, path-based trust inference such as Mui et al. (2002), Wang and Singh (2006), Liu et al. (2010), Hang et al. (2009), and Wang and Wu (2011) and component-based inference such as (Guha et al. 2004), Massa and Avesani (2005), Ziegler and Lausen (2005), and Zhou and Hwang (2007) all belong to trust inference algorithms. Although the main focus of this article is on the subjective metrics, our proposed subgraph extraction can also be applied to the objective trust metrics.

Despite the success of most existing inference algorithms, they share the scalability and usability limitations. To address these issues, we propose subgraph extraction for trust inference. The core of our subgraph extraction consists of two stages. The first stage, which serves for path-based trust inference, selects a set of paths from the trustor to the trustee. The second stage aims to produce a connected component between the trustor and the trustee for component-based trust inference. In addition, the second stage of our subgraph extraction produces a relatively small subgraph which can be clearly displayed and help the end user better understand the inference result.

We now formally define the subgraph extraction problem for trust inference. In accordance to the corresponding two stages, the problem is divided into two subproblems: *path selection problem* and *component induction problem*.

Definition 3 Path Selection Problem

Given: a weighted directed graph $G(V, E)$; two nodes $s, t \in V$; and an integer K

Find: a set C with K paths from s to t that minimizes the error function $f(C)$

Definition 4 Component Induction Problem

Given: a set C of paths from s to t and an integer N

Find: an induced component $H(V', E')$ with at most N edges that minimizes the error function $g(H)$, where $V' \subseteq \{v|(u,v)$ or $(v,u) \in P, P \in C\}$ and $E' \subseteq \{e|e \in P, P \in C\}$

We next discuss the error function in the definitions. The error function $f(C)$ in Definition 3 indicates the goodness of the extracted paths, and $f(C)$ reaches its minimum value when C contains all the possible paths from s to t. Similarly, the error function $g(H)$ in Definition 4 reaches its minimum value if $H = G$. In this article, we use *P*-error, which is defined as follows, as the error function for both subproblems, i.e., $f = g =$ P-error.

Definition 5 P-error

For a given trustor-trustee pair, the error function P-error is defined as

$$P\text{-error} = |p_{\text{sub}} - p_{\text{whole}}|,$$

where p_{sub} is the trustworthiness score inferred from the subgraph and p_{whole}, which serves as a ground truth, is the trustworthiness score inferred from the whole graph.

Path Selection

In the path selection stage, we aim to extract a few paths from the trustor to the trustee as an intermediate step to speed up path-based trust inference algorithms. These extracted paths will also serve as the input for the component induction stage.

There are two preprocessing steps in our extraction method. First of all, trust is interpreted as the probability by which the trustor expects that the trustee will perform a given action. This interpretation of trust is adopted by many existing trust inference algorithms, and it allows trust to be multiplicatively propagated along a path (Liu et al. 2010). Second, we transform probability into weight by negative logarithm. Namely, the local trust value on the edge e is interpreted as probability $p(e)$, and the probability $p(e)$ is transformed to weight $w(e) = -\log(p(e))$. Based on these two steps, the weight of a path P can be presented as

$$\begin{aligned} w(P) &= \sum_{e \in P} -\log(p(e)) = -\log(\prod_{e \in P} p(e)) \\ &= -\log(Pr(P)). \end{aligned}$$

As a result, finding a path of high trustworthiness in the original network is equivalent to finding a short path in the transformed network. We will use this transformed weighted graph $G(V, E)$ as the input of our method.

Then, the path selection problem becomes to extract top-k short paths from the trustor to the trustee in the transformed graph $G(V, E)$. Many existing algorithms can be plugged into this stage, such as Yen's k-shortest loopless paths (KS) (Yen 1971), and path sampling (PS) (Hintsanen et al.

Algorithm 1 KS algorithm (see the appendix for the details)

Input: Weighted directed graph $G(V, E)$, two nodes $s, t \in V$, and a parameter K of path number
Output: Set C with K paths from s to t

C = k-shortest(G, s, t, K)
return C

2010). In our experiments, we found that KS algorithm performs best even if the multiplicative property of the interpretation does not hold, and we therefore recommend KS in this stage. A brief skeleton of the KS algorithm is shown in Algorithm 1, and the detailed algorithms for KS and PS are presented in the appendix for completeness.

Algorithm Analysis

The worst-case time complexity of KS is $O(K|V|(|E| + |V|\log|V|))$, which is known as the best result to ensure that k-shortest loopless paths can be found in a directed graph (Hershberger et al. 2007). However, the actual wall-clock time of KS on many real graphs is often much better than such worst-case scenario (Martins and Pascoal 2003). In fact, based on our experiments, we find that it empirically scales near linearly wrt the graph size $|V|$ in the chosen datasets.

Component Induction

In the component induction, we take the output of path selection stage (i.e., a set of K paths) as input and output a small connected component from the trustor to the trustee. The output of the component induction stage not only acts as an intermediate step to speed up component-based trust inference algorithms but also helps to improve the usability of trust inference by interpreting the inference results for the end users. Notice that although our upcoming proposed algorithm EVO could also be applied on the whole graph, we do not recommend it in practice for the following two reasons: (1) most trustworthy paths have already been captured by the path selection stage (i.e., KS), and (2) applying EVO on the whole graph would cost more memory and time to achieve high accuracy. We will present more detailed experimental evaluations to validate this in the next section.

Algorithm 2 EVO algorithm

Input: Set C of paths from s to t and the directly induced component $G^c(V^c, E^c)$, as well as a constraint N of the edge number
Output: Induced component $H(V', E')$ with at most N edges

```
define 0/1 vector B of size |E^c| where each element
  in B stands for the existence of a corresponding edge
  in G^c
initialize m vectors S ← {B_1, B_2, ..., B_m}, with at
  most N 1-bits for each vector
while iter > 0 do
  for each vector B_i in S do
    repeat
      mutate B_i to B_{i+m} with mutation proba-
        bility 1/|E^c|
    until the number of 1-bits in B_{i+m} ≤ N
  end for
  compute P-error results for the 2m vectors
    {B_1, B_2, ..., B_{2m}}
  S ← the best m vectors from the 2m ones
  iter ← iter - 1
end while
B_final ← the best vector in S
return the corresponding component H(V', E') of
  B_final
```

In general, our proposed EVO algorithm (shown in Algorithm 2) belongs to the so-called evolutionary methods (Bäck 1996). It aims to minimize P-error under the constraint of edge number. The input component $G^c(V^c, E^c)$ is directly induced from the set C of paths from s to t, where $V^c = \{v|(u, v) \in P \text{ or } (v, u) \in P, P \in C\}$ and $E^c = \{e|e \in P, P \in C\}$. There are two implicit parameters in the algorithm, i.e., the initial vector number m and iteration number iter.

We now explain EVO in detail. The first step of EVO is to establish a one-to-one correspondence between the edges in G^c and the elements in vector B. Each element of B is a 0/1 bit where 1 indicates that the corresponding edge exists and 0 indicates otherwise. The vector has exactly $|E^c|$ bits where $|E^c|$ is the edge number of G^c. In the second step, the algorithm generates m vectors $B_1, B_2, \ldots, B_m$, and each of them has at most N 1-bits. In our implementation, we apply a constant-time search in C to find a subset of paths with minimized P-error. In the following steps, EVO adopts *mutation* on each of these vectors to separately generate m new vectors $B_{m+1}, B_{m+2}, \ldots, B_{2m}$. In the mutation from B_i to B_{i+m}, each bit of B_i is changed with probability $1/|E^c|$. If the resulting vector has more than N 1-bits, the mutation operation is redone. The error function, which is P-error in our case, is then computed on each of these $2m$ vectors, and the m vectors with smallest P-error are kept to the next iteration. For efficiency, the P-error computation on vector B herein means computing the P-error between $G^c(V^c, E^c)$ and the component corresponding to the vector B. Namely, we use the input component $G^c(V^c, E^c)$ as an approximation of the ground truth in this stage.

Algorithm Analysis

The time complexity of EVO is summarized in the following lemma, which basically says that the expected time complexity of EVO scales linearly wrt both initial vector number m and iteration number iter.

Lemma 1 *The average-case time complexity of EVO is $O(\text{iter} \cdot m(|E^c|/N + \theta))$, where θ is the time complexity of the error function computation.*

Proof In the mutation step of EVO, with mutation probability $1/|E^c|$, the expected number of bit changes is 1. This step is expected to be redone only when the number of 1-bits is N and the bit change is from 0 to 1. Under this condition, the probability of bit change from 0 to 1 is $(|E^c| - N)/|E^c|$. Therefore, the expected iteration number of the mutation step is $|E^c|/N$. Therefore, the whole expected time complexity of EVO is $O(\text{iter}(m \cdot |E^c|/N + m\theta)) = O(\text{iter} \cdot m(|E^c|/N + \theta))$, which completes the proof. □

Experimental Evaluation

In this section, we first describe the experimental setup and then present the results.

Experimental Setup

We first describe the datasets and the representatives of path-based and component-based trust inference algorithms. All algorithms are implemented in Java and have been run on a T400 ThinkPad with 1,280 m jvm heap space. Few other activities are done during the experiments.

Datasets Description

We use the advogato (http://www.trustlet.org/wiki/Advogato_dataset) datasets in our experiments, because advogato is a trust-based social network and it contains multilevel trust assertions. There are four levels of trust assertions in the network, i.e., "Observer," "Apprentice," "Journeyer," and "Master." These assertions can be mapped into real numbers in [0,1]. In our experiments, we map "Observer," "Apprentice," "Journeyer," and "Master" to 0.1, 0.4, 0.7, and 0.9, respectively. The statistics of the datasets is shown in Table 1.

Trust Inference Representatives

To evaluate our subgraph extraction method, we need to apply trust inference algorithms on the whole graph and on our extracted subgraph to compare their effectiveness and efficiency. We chose *CertProp* (Hang et al. 2009) as the representative of path-based inference algorithms, and *Appleseed* (Ziegler and Lausen 2005) as the representative of component-based inference algorithms.

P-error computation in CertProp needs to first compute the ground truth p_{whole} by finding all paths from the trustor to the trustee in the whole graph. This computation, however, easily causes the overflow of the jvm heap space even on the advogato-1 graph. Following the suggestions in the original CertProp (Hang et al. 2009), we apply the fixed search strategy and search all paths whose length is not longer than seven as an approximation of the ground truth. For CertProp, we define *collapsed samples* as the trustor-trustee pairs of which the P-error computation either exceeds the range of Java.lang.Double or runs out of the jvm heap space. We randomly select 100 node pairs out of 122 samples, where the rest 22 of them are collapsed samples. Our experimental results are all based on the average of these 100 samples. Notice that, as discussed in the path selection section, the multiplicative property of the probability interpretation does not hold in CertProp. As to Appleseed, we apply linear normalization on the outputs, since the algorithm can produce arbitrary trustworthiness scores.

Experimental Results

We now present the experimental results of our subgraph extraction method. In our experiments, the effectiveness, efficiency comparisons, and interpretation results are all based on the advogato-1 graph, as we found CertProp on the whole graph becomes computationally infeasible on all the other larger datasets. We evaluate the scalability of our method using all the datasets (i.e., advogato-1 to advogato-6). As for EVO, we set $m = 5$ and iter $= 10$ unless otherwise specified. The edge constraint N is set as $K/2$.

Effectiveness

For effectiveness, we first study how CertProp and Appleseed perform on the KS subgraph (the output of path selection stage) and EVO subgraph (the output of component induction stage), respectively. The results are shown in Fig. 1. We can observe that all the P-error values of CertProp and Appleseed are less than 0.04, indicating that our extracted subgraphs, which are based on a small set of carefully selected paths and an evolutionary strategy, provide high accuracy for the trust inference algorithms.

Remember that the proposed EVO is always applied on the output of the path selection stage (referred to as “EVO + KS”). Here, for comparison purpose, we also apply EVO on the entire graph (referred to as “EVO + whole graph”). With the same parameter setting, the results are shown in Fig. 2. It can be seen that EVO on KS outperforms EVO on the whole graph. The reason is as follows. As an evolutionary algorithm, EVO (either on KS or on the entire graph) finds a local minima. By restricting the search space to those highly trustworthy paths (i.e., the output of KS), it converges to a better local minima in terms of P-error.

Finally, to compare EVO with existing component induction algorithms, we implement the *Monte Carlo pruning (MC)* method (Hintsanen and Toivonen 2008) and the *proximity extraction (PE)* method (Koren et al. 2006). As mentioned in the historical background, MC is proposed for the reliable subgraph extraction problem. The key idea of MC is to measure a relevance score for each edge by Monte Carlo simulations and then remove the edges of lowest relevance scores. On the other hand, PE is proposed for the proximity computation problem where a small set of paths are selected to maximize the proposed proximity objective function. We plot the comparison results in Fig. 3. Again, we can see that EVO outperforms both MC and PE wrt P-error. In fact, MC induces a component by successively deleting edges (edge-level component induction), while PE only selects a smaller set of paths (path-level component induction). Our EVO algorithm outperforms MC and PE because EVO combines these two levels of component induction by searching a smaller set of paths in the initial step and then evolving the resulting component on the edge level.

Efficiency

First, we compare the different algorithmic choices in the path selection stage. To this end, we compare the wall-clock time of KS with an alternative path selection algorithm *path sampling (PS)* (Hintsanen et al. 2010). The results are shown in Fig. 4. Note that the y-axis is of log scale. As we can see from the figure, although PS is slightly faster than KS when $K = 5$, the wall-clock time of PS is much longer than that of KS when K is greater than 30. For example, the wall-clock time of PS is more than 170× longer than that of KS when $K = 100$. Therefore, we recommend using KS for path selection.

Next, we study the computational savings by applying the proposed subgraph extraction as the intermediate steps for the existing trust inference algorithms. To this end, we report the wall-clock time of CertProp on the output of the path selection stage and Appleseed on the output of the component induction stage, respectively. The results are shown in Fig. 5 where the y-axis is of log scale. Notice that the reported time includes the wall-clock time of both subgraph extraction and trust inference. In the figure, we also plot the wall-clock time of CertProp and Appleseed on the entire graph for comparison. We can see that our subgraph extraction method saves the wall-clock time for both path-based trust inference and component-based trust inference, especially for the former one. For example, when $K = 10$, our subgraph extraction method achieves up to 2,400× and 5.4× speedup for CertProp and Appleseed, respectively. Even when K grows to more than 60, our method can still achieve 200 – 400× speedup for CertProp.

Next, we compare the efficiency between applying EVO on KS and applying EVO on the whole graph. With $N = K/2$, the results are shown in Fig. 6. As we can see, the wall-clock time of EVO on KS (which includes the wall-clock time of both EVO and KS) is much faster than EVO on the whole graph. Together with the effectiveness results (Fig. 2), we recommend running EVO on the KS subgraph in practice.

Finally, we evaluate how the parameters m and iter in EVO affect the wall-clock time. In this experiment, we fix $K = 20$ and $N = 10$, and the results are shown in Fig. 7. We can observe that the wall-clock time of EVO scales linearly wrt iter for any fixed m, which is consistent with the time complexity analysis shown before.

Scalability

We now evaluate the scalability of our method on datasets advogato-1 to advogato-6. Figure 8 shows the results, where the y-axis is of log scale. In this experiment, we fix $K = 10$ and $N = 5$.

We can observe from the figure that even on the largest graph of 5,428 nodes and 51,293 edges, KS can help to infer the trustworthiness score within 25 s. In addition, KS scales near linearly wrt the underlying graph size. As to EVO, the wall-clock time stays stable in spite of the growth of the graph size. The reason is that $|E^c|$ scales near linearly to K due to many overlapping edges and N is set to $K/2$. Consequently, $|E^c|/N$ is close to a constant, and the time complexity of EVO can be approximated to $O(\text{iter} \cdot m \cdot \theta)$.

Usability/Interpretation

Another important goal of the proposed EVO is to improve the usability in trust inference by interpreting the inferred trustworthiness score for end users. An illustrative example is shown in Fig. 9. The whole graph and the induced KS subgraph by the path selection stage are also plotted for comparison.

From the figures, we can see that the whole graph is hard for interpretation. As to the KS subgraph, although the number of edges has significantly decreased compared with the original whole graph, there are still some redundant edges which might diverge end users' attention. On the other hand, the EVO subgraph only presents the most important participants and their trust opinions, providing a much clearer explanation on how the trustworthiness score is inferred.

Future Directions

On one hand, much of the research in trust inference focuses on the inference accuracy, while inference efficiency is also important in real-world trust inference applications, especially in those online applications. Future work should be able to find the best trade-offs between effectiveness and efficiency according to the specific applications. On the other hand, we believe that usability is becoming a new requirement for trust inference. Users start to care about not only who they should trust but also why they should trust. It is also interesting to incorporate distrust in the subgraph extraction as users may also concern about why they should not trust someone.

Acknowledgment

This work is supported by the National 863 Program of China (No. 2012AA011205), and the National Natural Science Foundation of

China (No. 91318301, 61021062, 61073030). The second author was partly sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053.

Appendix

To find K short paths from graph $G(V, E)$ in the path selection stage, many existing algorithms can be used. We consider two representative algorithms from the literature. Here, we present the detailed algorithm description for completeness.

The first algorithm is *Yen's k-shortest loopless paths (KS)* algorithm (Yen 1971), which is shown in Algorithm 3.

In the algorithm, we use Dijkstra's algorithm for finding a shortest path. All the computed paths are loopless by temporarily removing visited nodes. The key idea of the KS algorithm is *deviation*. The *deviation node* d of path P is the node that makes P deviate from existing paths in the candidate set C. For each node v between d (inclusive) and trustee t (exclusive) in P, the *deviated shortest path* from node v to t is computed by temporarily removing the edge starting at v in P. The computed deviated shortest path *post* and the subpath *pre* (the path from s to v in P) are combined to form a possible path candidate. For the nodes before d, possible shortest paths are already computed and included in X. Based on deviation, KS finds the K-shortest paths from trustor s to trustee t one by one. Following Martins and Pascoal's implementation (Martins and Pascoal 2003), we compute the deviated shortest path from deviation node d to the trustee in a reverse order.

The other algorithm is the randomized algorithm *path sampling (PS)* (Hintsanen et al. 2010), which is proposed for the *most reliable subgraph problem* (Hintsanen and Toivonen 2008). While PS is proposed for undirected graphs, trust relationships in social networks should be directed as trust is asymmetric in nature (Golbeck and Hendler 2006). Therefore, we adapt PS (as shown in Algorithm 4) for a directed graph.

PS considers the input graph as a Bernoulli random graph (Robins et al. 2007), and the algorithm is based on the *edge decision* of this random graph. An edge is randomly decided as true with probability $p(e)$, and a path is decided as true if all the edges on the path are decided as true. At the beginning of each iteration, all the edges of the graph are re-decided, and these

Algorithm 3 Detailed KS algorithm

Input: Weighted directed graph $G(V, E)$, two nodes $s, t \in V$, and a parameter K of path number
Output: Set C with K paths from s to t

```
X ← shortest path from s to t
C ← shortest path from s to t
while |C| < K and X ≠ Ø do
    P ← remove the shortest path in X
    d ← the deviation node of P
    for each node v between d (inclusive) and trustee
    t (exclusive) in P do
        pre ← subpath from trustor s to v in P
        post ← the deviated shortest path from v to t
        combine pre and post, and add it to X
    end for
    C ← C + the shortest path in X
end while
return C
```

Algorithm 4 PS algorithm

Input: Weighted directed graph $G(V, E)$, two nodes $s, t \in V$, and a parameter K of path number
Output: Set C with K paths from s to t

```
C ← shortest path from s to t
while |C| < K do
    re-decide all the edges in E
    for each path P in C do
        if P is decided as true then
            F ← F + P
        end if
    end for
    while F ≠ Ø do
        re-decide the most overlapped edge in F as
        failed
        remove failed paths from F, if there are any
    end while
    P ← the shortest path among the non-failed edges
    from s to t
    if P ≠ Ø then
        C ← C + P
    end if
end while
return C
```

graph decisions provide opportunities for distrust information to be contained. Like KS, PS first adds a shortest path into candidate set C. PS then tries to find a graph decision based on which none of the paths in C are true. To avoid the situation when this graph decision is hardly found, PS stores the true paths in C to a temporary set F and deliberately fails the most overlapping edges in F until none of the paths in F are true. Finally, based on the results of graph decision and edge failing, PS finds the shortest path P among the non-failed edges from trustor s to trustee t and adds it to C. The algorithm ends until K paths are found.

PS allows some distrust information to be incorporated into the extracted subgraph, which could in turn lower the P-error based on our experiments. However, the time complexity of PS is difficult to estimate, since the wall-clock time depends on the graph density. In addition, as shown in our experiments, the wall-clock time of PS is especially long when K becomes sufficiently large. We conjecture that PS can be used in dense graphs where numerous paths exist between node pairs.

Cross-References

- ▶ Computational Trust Models
- ▶ Modeling Trust and Reputation in Social Networks
- ▶ Trust in Social Network
- ▶ Trust Metrics and Reputation Systems
- ▶ Trust Network and Trust Inference

References

Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, New York

Barbian G (2011) Assessing trust by disclosure in online social networks. In: Proceedings of the international conference on advances in social networks analysis and mining, ASONAM '11, Kaohsiung, pp 163–170

Buchegger S, Le Boudec JY (2004) A robust reputation system for mobile ad-hoc networks. Technical report, KTH Royal Institute of Technology, Theoretical Computer Science Group

Faloutsos C, McCurley KS, Tomkins A (2004) Fast discovery of connection subgraphs. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, KDD '04, Seattle, pp 118–127

Golbeck J, Hendler J (2006) Inferring binary trust relationships in web-based social networks. ACM Trans Internet Technol 6:497–529

Guha R, Kumar R, Raghavan P, Tomkins A (2004) Propagation of trust and distrust. In: Proceedings of the 13th international conference on world wide web, WWW '04, New York. ACM, pp 403–412

Hang CW, Wang Y, Singh MP (2009) Operators for propagating trust and their evaluation in social networks. In: Proceedings of the 8th international conference on autonomous agents and multiagent systems, AAMAS '09, Budapest, vol 2. International Foundation for Autonomous Agents and Multiagent Systems, pp 1025–1032

Hershberger J, Maxel M, Suri S (2007) Finding the k shortest simple paths: a new algorithm and its implementation. ACM Trans Algorithms 3(4):45

Hintsanen P, Toivonen H (2008) Finding reliable subgraphs from large probabilistic graphs. Data Mini Knowl Discov 17(1):3–23

Hintsanen P, Toivonen H, Sevon P (2010) Fast discovery of reliable subnetworks. In: Proceedings of the international conference on advances in social networks analysis and mining, ASONAM '10, Odense, pp 104–111

Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on world wide web, WWW '03, Budapest. ACM, pp 640–651

Koren Y, North S, Volinsky C (2006) Measuring and extracting proximity in networks. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '06, Philadelphia. ACM, pp 245–255

Liu G, Wang Y, Orgun M (2010) Optimal social trust path selection in complex social networks. In: Proceedings of the twenty-fourth AAAI conference on artificial intelligence, AAAI '10, Atlanta, pp 1391–1398

Martins E, Pascoal M (2003) A new implementation of Yen's ranking loopless paths algorithm. 4OR Q J Oper Res 1(2):121–133

Massa P, Avesani P (2005) Controversial users demand local trust metrics: an experimental study on epinions.com community. In: Proceedings of the AAAI conference on artificial intelligence, AAAI '05, Pittsburgh, pp 121–126

Mui L, Mohtashemi M, Halberstadt A (2002) A computational model of trust and reputation. In: Proceedings of the 35th annual Hawaii international conference on system sciences, HICSS '02, Big Island. IEEE, pp 2431–2439

Nordheimer K, Schulze T, Veit D (2010) Trustworthiness in networks: a simulation approach for approximating local trust and distrust values. In: Trust management IV, Morioka. IFIP advances in information and communication technology, vol 321. Springer, Boston, pp 157–171

Richardson M, Agrawal R, Domingos P (2003) Trust management for the semantic web. In: The semantic web, Sanibel Island. Lecture notes in computer science, vol 2870. Springer, Berlin/Heidelberg, pp 351–368

Robins G, Pattison P, Kalish Y, Lusher D (2007) An introduction to exponential random graph (p*) models for social networks. Soc Netw 29(2):173–191

Tong H, Faloutsos C, Koren Y (2007) Fast direction-aware proximity for graph mining. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, KDD ’07, San Jose. ACM, pp 747–756

Wang Y, Singh MP (2006) Trust representation and aggregation in a distributed agent system. In: Proceedings of the AAAI conference on artificial intelligence, AAAI ’06, Boston, pp 1425–1430

Wang Y, Singh MP (2007) Formal trust model for multi-agent systems. In: Proceedings of the 20th international joint conference on artificial intelligence, IJCAI ’07, Hyderabad, pp 1551–1556

Wang G, Wu J (2011) Multi-dimensional evidence-based trust management with multi-trusted paths. Future Gener Comput Syst 27(5):529–538

Xiong L, Liu L (2004) Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. IEEE Trans Knowl Data Eng 16(7):843–857

Yao Y, Zhou J, Han L, Xu F, Lü J (2011) Comparing linkage graph and activity graph of online social networks. In: Social informatics, Singapore. Lecture notes in computer science, vol 6984. Springer, Berlin/Heidelberg, pp 84–97

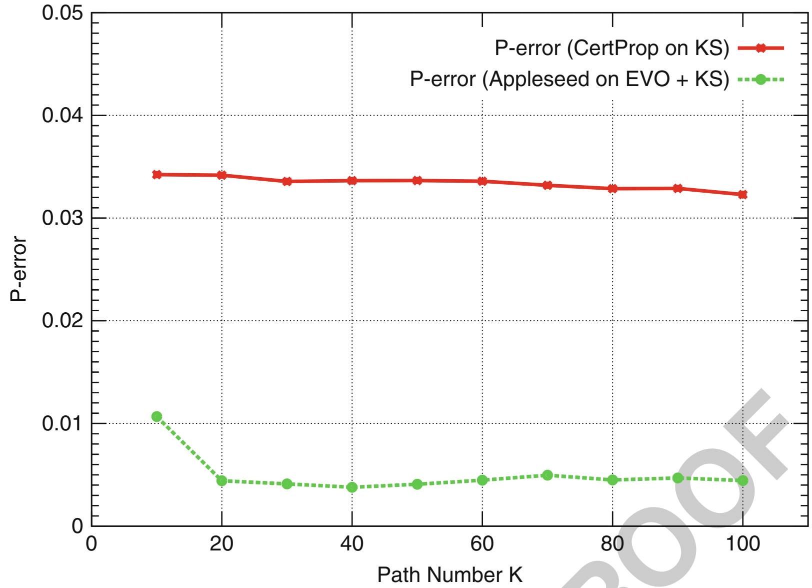
Yen J (1971) Finding the k shortest loopless paths in a network. Manag Sci 17(11):712–716

Zhou R, Hwang K (2007) Powertrust: a robust and scalable reputation system for trusted peer-to-peer computing. IEEE Trans Parallel Distrib Syst 18(4):460–473

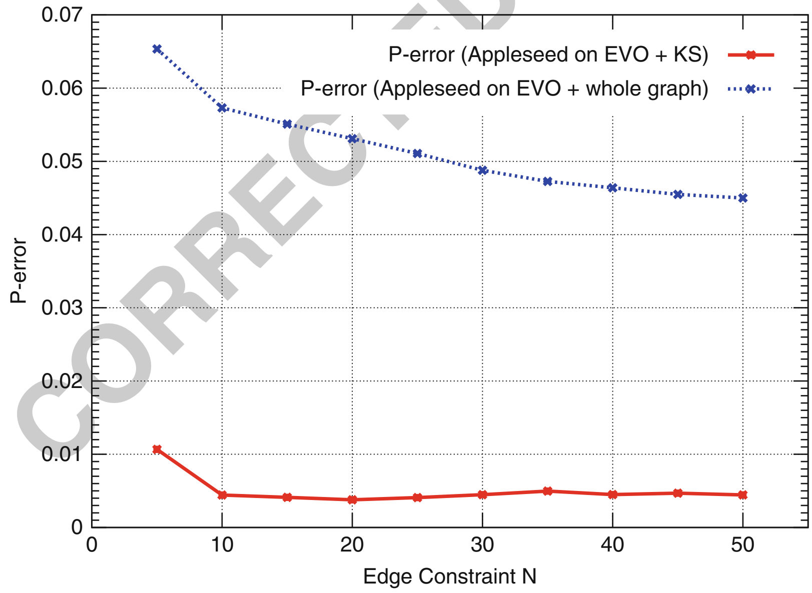
Ziegler C, Lausen G (2005) Propagation models for trust and distrust in social networks. Inf Syst Front 7(4):337–358

Subgraph Extraction for Trust Inference in Social Networks, Table 1 High-level statistics of advogato datasets

Graph	Nodes	Edges	Avg. degree	Avg. clustering	Avg. diameter	Date
Advogato-1	279	2,109	15.1	0.45	4.62	2000-02-05
Advogato-2	1,261	12,176	19.3	0.36	4.71	2000-07-18
Advogato-3	2,443	22,486	18.4	0.31	4.67	2001-03-06
Advogato-4	3,279	32,743	20.0	0.33	4.74	2002-01-14
Advogato-5	4,158	41,308	19.9	0.33	4.83	2003-03-04
Advogato-6	5,428	51,493	19.0	0.31	4.82	2011-06-23

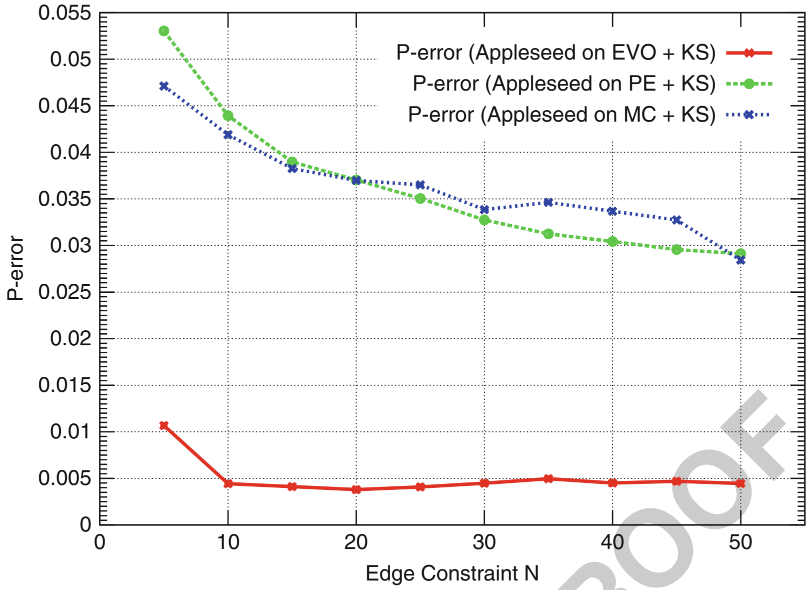


Subgraph Extraction for Trust Inference in Social Networks, Fig. 1 Effectiveness of our subgraph extraction method with edge number constraint $N = K/2$. In all cases, the P-error is less than 0.04

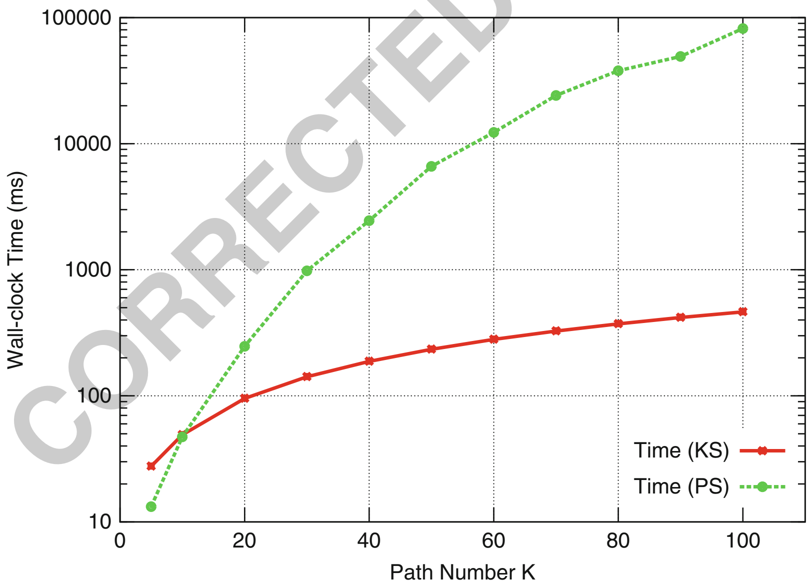


Subgraph Extraction for Trust Inference in Social Networks, Fig. 2 Comparison of EVO on KS vs. EVO on the whole graph with edge number constraint $N = K/2$. EVO on KS outperforms EVO on the whole graph

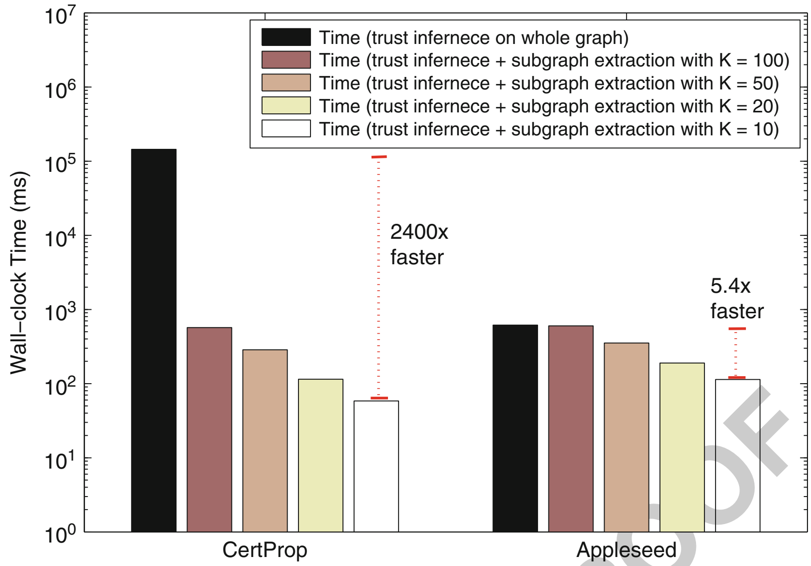
ESN 357 — 2013/11/4 — 13:39 — page 14 — #16



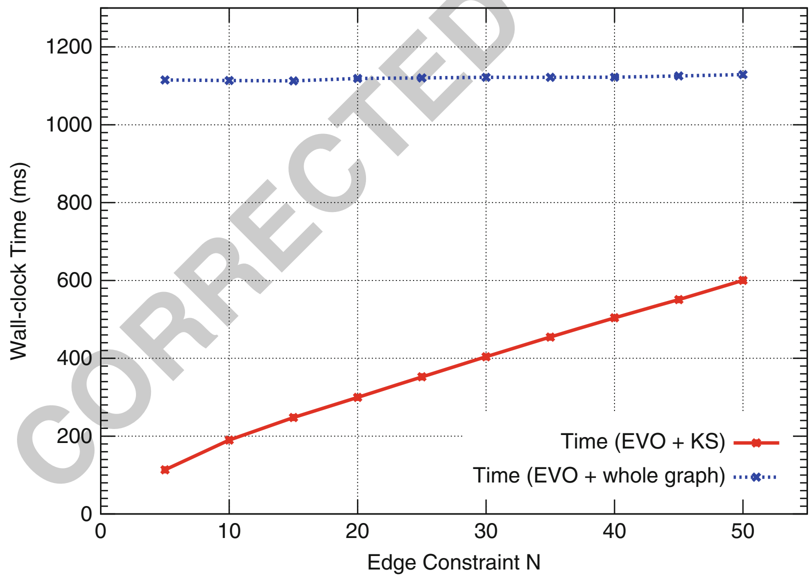
Subgraph Extraction for Trust Inference in Social Networks, Fig. 3 Comparison of different component induction algorithms with edge number constraint $N = K/2$. EVO outperforms the existing component induction algorithms



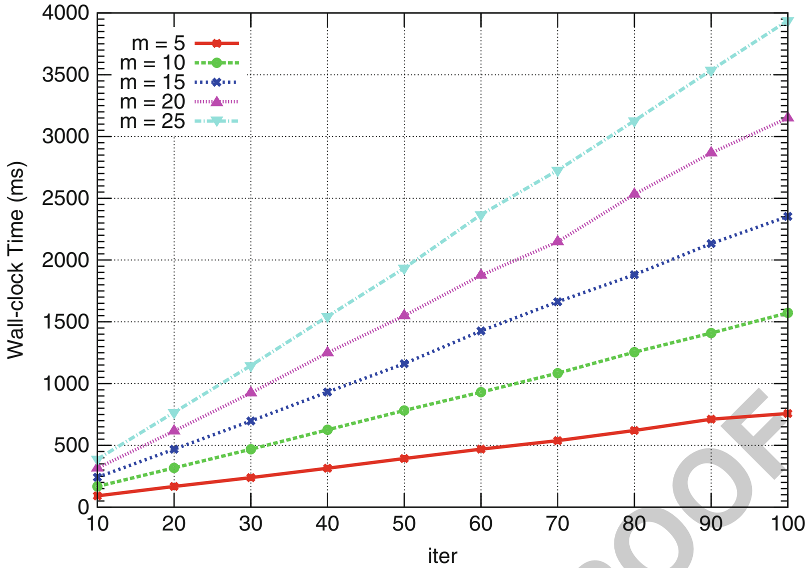
Subgraph Extraction for Trust Inference in Social Networks, Fig. 4 The average wall-clock time of KS and PS. The average wall-clock time of KS is much faster than that of PS when K is greater than 30



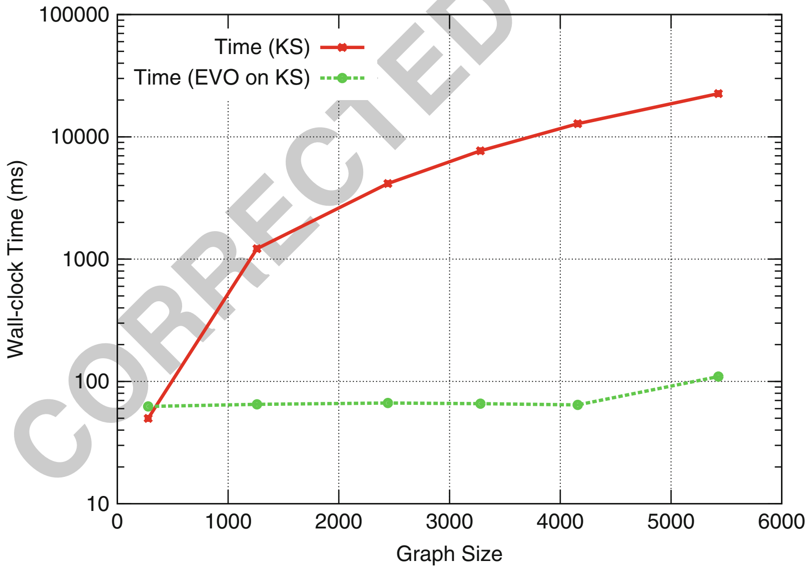
Subgraph Extraction for Trust Inference in Social Networks, Fig. 5 The average wall-clock time of CertProp on KS and Appleseed on KS + EVO. We achieve up to 2,400× speedup



Subgraph Extraction for Trust Inference in Social Networks, Fig. 6 The average wall-clock time of EVO on KS and EVO on the whole graph with edge number constraint $N = K/2$. EVO on KS is much faster



Subgraph Extraction for Trust Inference in Social Networks, Fig. 7 The average wall-clock time of EVO with $K = 20$ and $N = 10$. EVO scales linearly wrt iter for the fixed m



Subgraph Extraction for Trust Inference in Social Networks, Fig. 8 The scalability of our subgraph extraction method. KS scales near linearly wrt the graph size, while the wall-clock time of EVO stays almost constant

Subgraph Extraction for Trust Inference in Social Networks, Fig. 9 The interpretation example of the whole graph, KS-20, and EVO-10 on KS-20. (**a**) The original whole graph. (**b**) KS subgraph with $K = 20$. The paths are from "Adrian" (the leftmost node) to "terop" (the rightmost node). (**c**) EVO subgraph with $N = 10$ on KS-20. The component is from "Adrian" to "terop"