NetProbe: A Fast and Scalable System for Fraud Detection in Online Auction Networks

Shashank Pandit, Duen Horng Chau, Samuel Wang, Christos Faloutsos *
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{shashank, dchau, samuelwang, christos}@cs.cmu.edu

ABSTRACT

Given a large online network of online auction users and their histories of transactions, how can we spot anomalies and auction fraud? This paper describes the design and implementation of NetProbe, a system that we propose for solving this problem. NetProbe models auction users and transactions as a *Markov Random Field* tuned to detect the suspicious patterns that fraudsters create, and employs a *Belief Propagation* mechanism to detect likely fraudsters. Our experiments show that NetProbe is both efficient and effective for fraud detection. We report experiments on synthetic graphs with as many as 7,000 nodes and 30,000 edges, where NetProbe was able to spot fraudulent nodes with over 90% precision and recall, within a matter of seconds. We also report experiments on a real dataset crawled from eBay, with nearly 700,000 transactions between more than 66,000 users, where NetProbe was highly effective at unearthing hidden networks of fraudsters, within a realistic response time of about 6 minutes. For scenarios where the underlying data is dynamic in nature, we propose *Incremental NetProbe*, which is an approximate, but fast, variant of NetProbe. Our experiments prove that Incremental NetProbe executes nearly doubly fast as compared to NetProbe, while retaining over 99% of its accuracy.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms

Keywords

Fraud detection, Bipartite cores, Markov random fields, Belief propagation

*This material is based upon work supported by the National Science Foundation under Grants No. IIS-0326322 IIS-0534205. This work is also supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA), an IBM Faculty Award, a Yahoo Research Alliance Gift, with additional funding from Intel, NTT and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

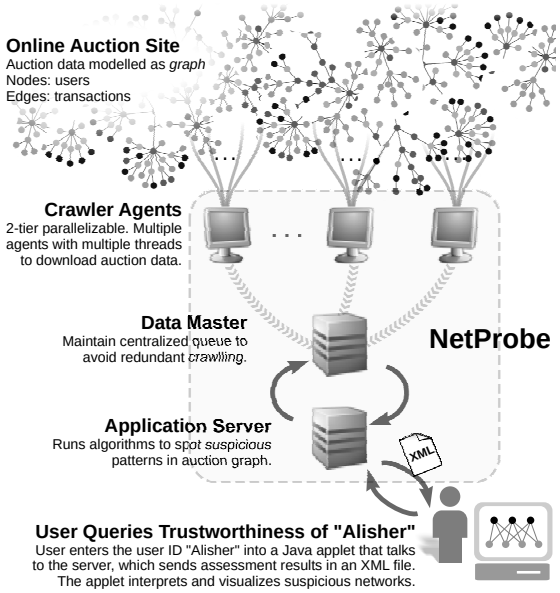

Figure 1: Overview of the NetProbe system

1. INTRODUCTION

Online auctions have been thriving as a business over the past decade. People from all over the world trade goods worth millions of dollars every day using these virtual marketplaces. EBay[1], the world's largest auction site, reported a third quarter revenue of $1,449 billion, with over 212 million registered users [6]. These figures represent a 31% growth in revenue and 26% growth in the number of registered users over the previous year. Unfortunately, rapid commercial success has made auction sites a lucrative medium for committing fraud. For more than half a decade, auction fraud has been the most prevalent Internet crime. Auction fraud represented 63% of the complaints received by the Federal Internet Crime Complaint Center last year. Among all the monetary losses reported, auction fraud accounted for 41%, with an average loss of $385 [10].

Despite the prevalence of auction frauds, auctions sites have not come up with systematic approaches to expose fraudsters. Typically, auction sites use a reputation based framework for aiding users to assess the trustworthiness of each other. However, it is not difficult for a fraudster to

[1] `http://www.ebay.com`

manipulate such reputation systems. As a result, the problem of auction fraud has continued to worsen over the past few years, causing serious concern to auction site users and owners alike.

We therefore ask ourselves the following research questions - given a large online auction network of auction users and their histories of transactions, how do we spot fraudsters? How should we design a system that will carry out fraud detection on auction sites in a fast and accurate manner?

In this paper, we propose NetProbe a system for fraud detection in online auction sites (Figure 1). NetProbe is a system that systematically analyzes transactions within users of auction sites to identify suspicious networks of fraudsters. NetProbe allows users of an online auction site to query the trustworthiness of any other user, and offers an interface to visually explains the query results. In particular, we make the following contributions through NetProbe:

- First, we propose data models and algorithms based on Markov Random Fields and belief propagation to uncover suspicious networks hidden within an auction site. We also propose an incremental version of NetProbe which performs almost twice as fast in dynamic environments, with negligible loss in accuracy.
- Second, we demonstrate that NetProbe is fast, accurate, and scalable, with experiments on large synthetic and real datasets. Our synthetic datasets contained as many as 7,000 users with over 30,000 transactions, while the real dataset (crawled from eBay) contains over 66,000 users and nearly 800,000 transactions.
- Lastly, we share the non-trivial design and implementation decisions that we made while developing NetProbe. In particular, we discuss the following contributions: (a) a parallelizable crawler that can efficiently crawl data from auction sites, (b) a centralized queuing mechanism that avoids redundant crawling, (c) fast, efficient data structures to speed up our fraud detection algorithm, and (d) a user interface that visually demonstrates the suspicious behavior of potential fraudsters to the end user.

The rest of this paper is organized as follows. We begin by reviewing related work in Section 2. Then, we describe the algorithm underlying NetProbe in Section 3 and explain how it uncovers dubious associations among fraudsters. We also discuss the incremental variant of NetProbe in this section. Next, in Section 4, we report experiments that evaluate NetProbe (as well as its incremental variant) on large real and synthetic datasets, demonstrating NetProbe's effectiveness and scalability. In Section 5, we describe NetProbe's full system design and implementation details. Finally, we summarize our contributions in Section 6 and outline directions for future work.

2. RELATED WORK

In this section, we survey related approaches for fraud detection in auction sites, as well as the literature on reputation systems that auction sites typically use to prevent fraud. We also look at related work on trust and authority propagation, and graph mining, which could be applied to the context of auction fraud detection.

Efforts from the Mass. In the past, attempts have been made to help people identify potential fraudsters. However, most of them are "common sense" approaches, recommended by a variety of authorities such as newspapers articles [20], law enforcement agencies [8], or even from auction sites themselves [7]. These approaches usually suggest that people be cautious at their end and perform background checks of sellers that they wish to transact with. Such suggestions however, require users to maintain constant vigilance and spend a considerable amount of time and effort in investigating potential dealers before carrying out a transaction.

To overcome this difficulty, self-organized vigilante organizations are formed, usually by auction fraud victims themselves, to expose fraudsters and report them to law enforcement agencies [1]. Unfortunately, such grassroot efforts are insufficient for regulating large-scale auction fraud, motivating the need for a more systematic approach to solve the auction fraud problem.

Auction Fraud and Reputation Systems. Reputation systems are used extensively by auction sites to prevent fraud. But they are usually very simple and can be easily foiled. In an overview, Resnick et al. [17] summarized that modern reputation systems face many challenges which include the difficulty to elicit honest feedback and to show faithful representations of users' reputation. Despite their limitations, reputation systems have had a significant effect on how people buy and sell. Melnik et al. [13] and Resnick et al. [18] conducted empirical studies which showed that selling prices of goods are positively affected by the seller's reputation, implying people feel more confident to buy from trustworthy sources. In summary, reputation systems might not be an effective mechanism to prevent fraud because fraudsters can easily trick these systems to manipulating their own reputation.

Chua et al. [5] have categorized auction fraud into different types, but they did not formulate methods to combat them. They suggest that an effective approach to fight auction fraud is to allow law enforcement and auction sites to join forces, which unfortunately can be costly from both monetary and managerial perspectives.

In our previous work, we explored a classification-based fraud detection scheme [3]. We extracted features from auction data to capture fluctuations in sellers' behaviors (e.g., selling numerous expensive items after selling very few cheap items). This method, though promising, warranted further enhancement because it did not take into account the patterns of interaction employed by fraudsters while dealing with other auction users. To this end, we suggested a fraud detection algorithm by identifying suspicious networks amongst auction site users [4]. However, the experiments were reported over a tiny dataset, while here we report an in-depth evaluation over large synthetic and real datasets, along with fast, incremental computation techniques.

Trust and Authority Propagation. Authority propagation, an area closely related to fraud detection, has been studied extensively in the context of Web search. PageRank [2] and HITS [11] treat a Web page as "important" if other "important" pages point to it. In effect, they propagate the importance of pages over hyperlinks connecting them. Trust propagation was used by TrustRank [9] to detect Web spam. Here, the goal was to distinguish between "good" and "bad" sites (e.g, phishers, sites with adult con-

tent, etc). Also related is the work by Neville et al. [14, 15], which aggregates features across nodes in a graph for classification of movie and stock databases. None of these techniques however, explicitly focuses on fraud detection.

Graph Mining: Remotely related is the work on graph mining, with (fascinating) discoveries about the Web graph topology [12], Internet topology [19], and fast algorithms to search and mine for specific, or frequent graph patterns (e.g., *gSpan* [22], the GraphMiner system [21] and related algorithms [16, 23, 25]). None of these techniques focuses on a systematic way to do large-scale, online auction fraud detection, which is the focus of our work.

3. NetProbe: PROPOSED ALGORITHMS

In this section, we present NetProbe's algorithm for detecting networks of fraudsters in online auctions. The key idea is to infer properties for a user based on properties of other related users. In particular, given a graph representing interactions between auction users, the likelihood of a user being a fraudster is inferred by looking at the behavior of its immediate neighbors . This mechanism is effective at capturing fraudulent behavioral patterns, and affords a fast, scalable implementation (see Section 4).

We begin by describing the *Markov Random Field* (MRF) model, which is a powerful way to model the auction data in graphical form. We then describe the *Belief Propagation* algorithm, and present how NetProbe uses it for fraud detection. Finally, we present an incremental version of NetProbe which is a quick and accurate way to update beliefs when the graph topology changes.

3.1 The Markov Random Field Model

MRFs are a class of graphical models particularly suited for solving inference problems with uncertainty in observed data. MRFs are widely used in image restoration problems wherein the observed variables are the intensities of each pixel in the image, while the inference problem is to identify high-level details such as objects or shapes.

A MRF consists of an undirected graph, each node of which can be in any of a finite number of states. The state of a node is assumed to statistically depend only upon each of its neighbors, and independent of any other node in the graph [2]. The dependency between a node and its neighbors is represented by a *Propagation Matrix* (ψ), where $\psi(i, j)$ equals the probability of a node being in state j given that it has a neighbor in state i.

Given a particular assignment of states to the nodes in a MRF, a likelihood of observing this assignment can be computed using the propagation matrix. Typically, the problem is to infer the maximum likelihood assignment of states to nodes, where the correct states for some of the nodes are possibly known before hand. Naive computation through enumeration of all possible state assignments is exponential in time. Further, there is no method known which can be theoretically proved to solve this problem for a general MRF. Therefore, in practice, the above problem is solved through heuristic techniques. One particularly powerful method is the iterative message passing scheme of belief propagation. This method, although provably correct only for a restricted class of MRFs, has been shown to perform extremely well for general MRFs occurring in a wide variety of disciplines (e.g., error correcting codes, image restoration, factor graphs, and particle physics [3]). Next, we describe how belief propagation solves the above inference problem for general MRFs.

[2]The general MRF model is much more expressive than discussed here. For a more comprehensive discussion, see [24].

Symbol	Definition
S	set of possible states
$\mathbf{b}_i(\sigma)$	belief of node i in state σ
$\psi(i, j)$	$(i, j)^{th}$ entry of the Propagation Matrix
$\mathbf{m}_{ij}$	message sent by node i to node j

Table 1: Symbols and definitions

3.2 The Belief Propagation Algorithm

As mentioned before, belief propagation is an algorithm used to infer the maximum likelihood state probabilities of nodes in a MRF, given a propagation matrix and possibly a prior state assignment for some of the nodes. In this section, we describe how the algorithm operates over general MRFs.

We denote vectors in bold font, and scalars in normal font. For any vector $\mathbf{v}$, $\mathbf{v}(k)$ denotes its k^{th} component. The set of possible states a node can be in is represented by S. For a node n, the probability of n being in state σ is called the *belief* of n in state σ, and is denoted by $\mathbf{b}_n(\sigma)$. Table 1 lists the symbols and their definitions used in this section.

Belief propagation functions via iterative message passing between nodes in the network. Let $\mathbf{m}_{ij}$ denote the message that node i passes to node j. $\mathbf{m}_{ij}$ represents i's opinion about the belief of j. At every iteration, each node i computes its belief based on messages received from its neighbors, and uses the propagation matrix to transform its belief into messages for its neighbors. Mathematically,

$$\mathbf{m}_{ij}(\sigma) \leftarrow \sum_{\sigma'} \psi(\sigma', \sigma) \prod_{n \in N(i) \setminus j} \mathbf{m}_{ni}(\sigma') \quad (1)$$

$$\mathbf{b}_i(\sigma) \leftarrow k \prod_{j \in N(i)} \mathbf{m}_{ji}(\sigma) \quad (2)$$

where $\mathbf{m}_{ij}$ is the message vector sent by node i to j
$N(i)$ is the set of nodes neighboring i
k is a normalization constant

Starting with a suitable prior on the beliefs of the nodes, belief propagation proceeds by iteratively passing messages between nodes based on previous beliefs, and updating beliefs based on the passed messages [4]. The iteration is stopped when the beliefs converge (within some threshold), or a maximum limit for the number of iterations is exceeded. Although convergence is not guaranteed theoretically, in practice the algorithm has been observed to converge quickly to reasonably accurate solutions.

3.3 NetProbe for Online Auctions

We now describe how NetProbe utilizes the MRF modeling to solve the fraud detection problem.

[3]For an excellent discussion on belief propagation and its generalizations to various problems, see [24].

[4]In case there is no prior knowledge available, each node is initialized to an unbiased state (i.e., it is equally likely to be in any of the possible states), and the initial messages are computed by multiplying the propagation matrix with these initial, unbiased beliefs.

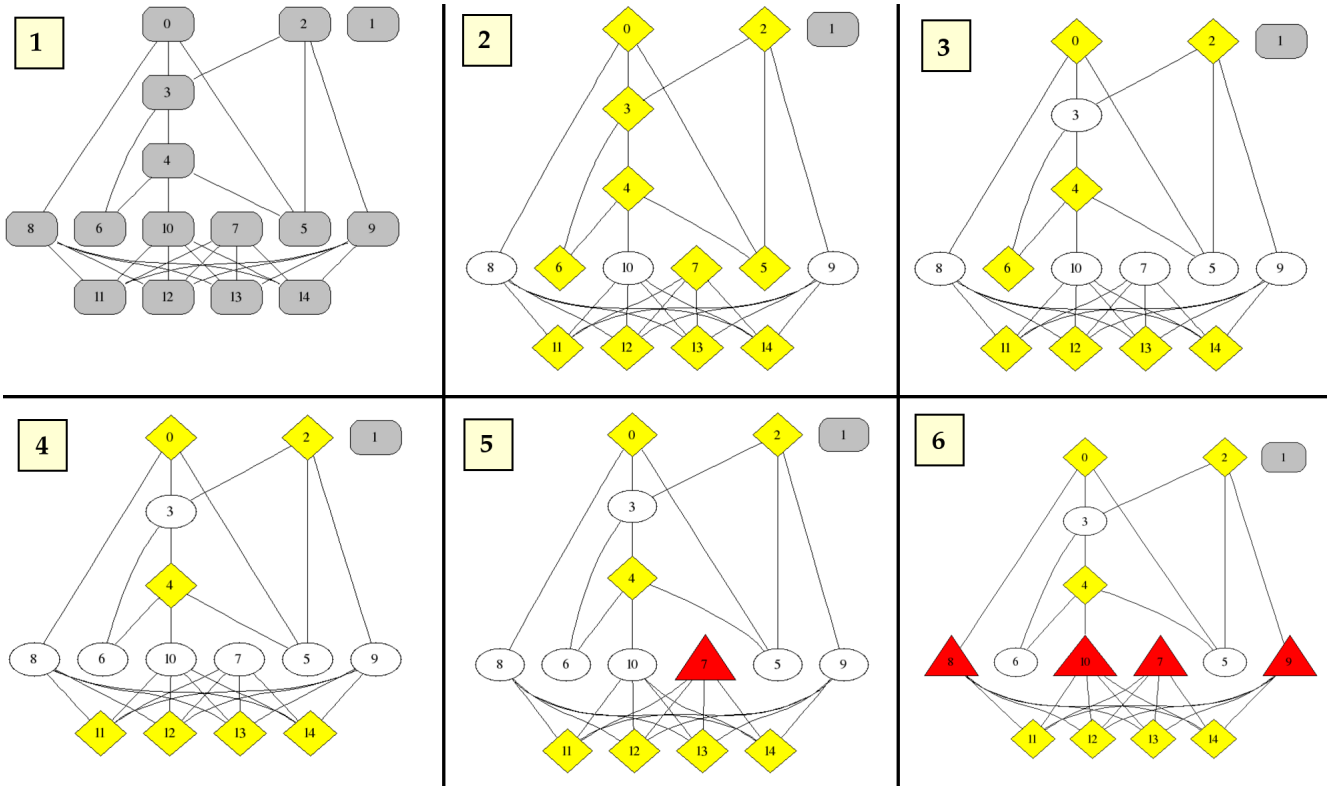

Figure 2: A sample execution of NetProbe. Red triangles represent fraudsters, yellow diamonds represent accomplices, white ellipses represent honest nodes, while gray rounded rectangles represent unbiased nodes.

Transactions between users are modeled as a graph, with a node for each user and an edge for one (or more) transactions between two users. As is the case with hyper-links on the Web (where PageRank [2] posits that a hyper-link confers authority from the source page to the target page), an edge between two nodes in an auction network can be assigned a definite semantics, and can be used to propagate properties from one node to its neighbors. For instance, an edge can be interpreted as an indication of similarity in behavior — honest users will interact more often with other honest users, while fraudsters will interact in small cliques of their own (to mutually boost their credibility). This semantics is very similar in spirit to that used by TrustRank [9], a variant of PageRank used to combat Web spam. Under this semantics, honesty/fraudulence can be propagated across edges and consequently, fraudsters can be detected by identifying relatively small and densely connected subgraphs (near cliques).

However, our previous work [4] suggests that fraudsters do not form such cliques. There are several reasons why this might be so:

- Auction sites probably use techniques similar to the one outlined above to detect probable fraudsters and void their accounts.
- Once a fraud is committed, an auction site can easily identify and void the accounts of other fraudsters involved in the clique, destroying the "infrastructure" that the fraudster had invested in for carrying out the fraud. To carry out another fraud, the fraudster will have to re-invest efforts in building a new clique.

Instead, we uncovered a different *modus operandi* for fraudsters in auction networks, which leads to the formation of *near bipartite cores*. Fraudsters create two types of identities and arbitrarily split them into two categories – *fraud* and *accomplice*. The fraud identities are the ones used eventually to carry out the actual fraud, while the accomplices exist only to help the fraudsters carry out their job by boosting their feedback rating. Accomplices themselves behave like perfectly legitimate users and interact with other honest users to achieve high feedback ratings. On the other hand, they also interact with the fraud identities to form near bipartite cores, which helps the fraud identities gain a high feedback rating. Once the fraud is carried out, the fraud identities get voided by the auction site, but the accomplice identities linger around and can be reused to facilitate the next fraud.

We model the auction users and their mutual transactions as a MRF. A node in the MRF represents a user, while an edge between two nodes denotes that the corresponding users have transacted at least once. Each node can be in any of 3 states — fraud, accomplice, and honest.

To completely define the MRF, we need to instantiate the propagation matrix. Recall that an entry in the propagation matrix $\psi(\sigma, \sigma')$ gives the likelihood of a node being in state σ' given that it has a neighbor in state σ. A sample instantiation of the propagation matrix is shown in Table 2. This instantiation is based on the following intuition: a fraudster tends to heavily link to accomplices but avoids linking to other bad nodes; an accomplice tends to link to both fraudsters and honest nodes, with a higher affinity for

Neighbor state	Node state		
	Fraud	Accomplice	Honest
Fraud	ϵ_p	$1-2\epsilon_p$	ϵ_p
Accomplice	0.5	$2\epsilon_p$	$0.5-2\epsilon_p$
Honest	ϵ_p	$(1-\epsilon_p)/2$	$(1-\epsilon_p)/2$

Table 2: Instantiation of the propagation matrix for fraud detection. Entry (i, j) denotes the probability of a node being in state j given that it has a neighbor in state i.

fraudsters; a honest node links with other honest nodes as well as accomplices (since an accomplice effectively appears to be honest to the innocent user.) In our experiments, we set ϵ_p to 0.05. Automatically learning the correct value of ϵ_p as well as the form of the propagation matrix itself would be valuable future work.

3.4 NetProbe: A Running Example

In this section, we present a running example of how NetProbe detects bipartite cores using the propagation matrix in Table 2. Consider the graph shown in Figure 2. The graph consists of a bipartite core (nodes $7, 8, \ldots, 14$) mingled within a larger network. Each node is encoded to depict its state — red triangles indicate fraudsters, yellow diamonds indicate accomplices, white ellipses indicate honest nodes, while gray rounded rectangles indicate unbiased nodes (i.e., nodes equally likely to be in any state.)

Each node is initialized to be unbiased, i.e., it is equally likely to be fraud, accomplice or honest. The nodes then iteratively pass messages and affect each other's beliefs. Notice that the particular form of the propagation matrix we use assigns a higher chance of being an accomplice to every node in the graph at the end of the first iteration. These accomplices then force their neighbors to be fraudsters or honest depending on the structure of the graph. In case of bipartite cores, one half of the core is pushed towards the fraud state, leading to a stable equilibrium. In the remaining graph, a more favorable equilibrium is achieved by labeling some of the nodes as honest.

At the end of execution, the nodes in the bipartite core are neatly labeled as fraudsters and accomplices. The key idea is the manner in which accomplices force their partners to be fraudsters in bipartite cores, thus providing a good mechanism for their detection.

3.5 Incremental NetProbe

In a real deployment of NetProbe, the underlying graph corresponding to transactions between auction site users, would be extremely dynamic in nature, with new nodes (i.e., users) and edges (i.e., transactions) being added to it frequently. In such a setting, if one expects an exact answer from the system, NetProbe would have to propagate beliefs over the entire graph for every new node/edge that gets added to the graph. This would be infeasible in systems with large graphs, and especially for online auction sites where users expect interactive response times.

Intuitively, addition of a few edges should not perturb the remaining graph by a lot (especially disconnected components.) To avoid wasteful recomputation of node beliefs from scratch, we developed a mechanism called *Incremental NetProbe*, which incrementally update beliefs of nodes upon small changes in the graph structure,

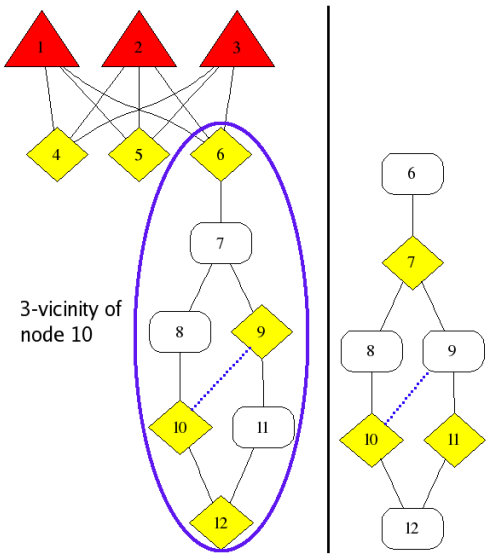

Figure 3: An example of Incremental NetProbe. Red triangles represent fraudsters, yellow diamonds represent accomplices, white ellipses represent honest nodes, while gray rounded rectangles represent unbiased nodes. An edge (shown as a dotted blue line) is added between nodes 9 and 10 of the graph on the left hand side. Normal propagation of beliefs in the 3-vicinity of node 10 (shown on the right hand side) leads to incorrect inference, and so nodes on the boundary of the 3-vicinity (i.e. node 6) should retain their beliefs.

The motivation behind Incremental NetProbe is that addition of a new edge will at worst result in minor changes in the immediate neighborhood of the edge, while the effect will not be strong enough to propagate to the rest of the graph. Whenever a new edge gets added to the graph, the algorithm proceeds by performing a breadth-first search of the graph from one of the end points (call it n) of the new edge, up to a fixed number of hops h, so as to retrieve a small subgraph, which we refer to as the *h-vicinity* of n. It is assumed that only the beliefs of nodes within the h-vicinity are affected by addition of the new edge. Then, "normal" belief propagation is performed only over the h-vicinity, with one key difference. While passing messages between nodes, beliefs of the nodes on the boundary of the h-vicinity are kept fixed to their original values. This ensures that the belief propagation takes into account the *global* properties of the graph, in addition to the *local* properties of the h-vicinity.

The motivation underlying Incremental NetProbe's algorithm is exemplified in Figure 3. The initial graph is shown on the left hand side, to which an edge is added between nodes 9 and 10. The 3-vicinity of node 10 is shown on the right hand side. The nodes on the right hand side are colored according to their inferred states based on propagating beliefs only in the subgraph *without* fixing the belief of node 6 to its original value. Note that the 3-vicinity does not capture the fact that node 6 is a part of a bipartite core. Hence the beliefs inferred are influenced only by the local structure of the 3-vicinity and are "out of sync" with the remaining graph. In order to make sure that Incremental NetProbe retains global properties of the graph, it is essential to fix

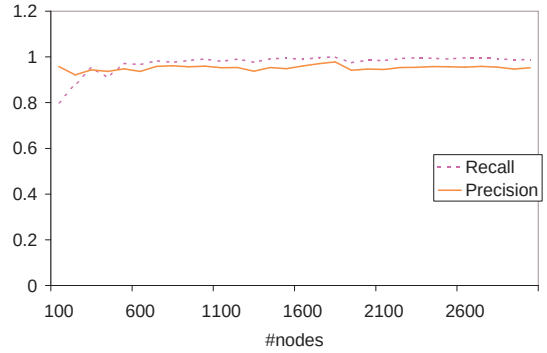

Figure 4: Accuracy of NetProbe over synthetic graphs with injected bipartite cores

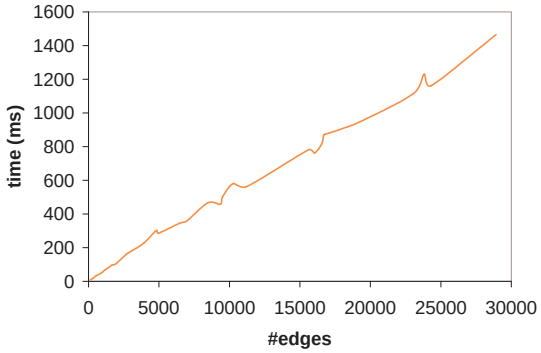

Figure 6: Scalability of NetProbe over synthetic graphs

the beliefs of nodes at the boundary of the 3-vicinity to their original values.

4. EVALUATION

We evaluated the performance of NetProbe over synthetic as well as real datasets. Overall, NetProbe was *effective* – it detected bipartite cores with very high accuracy – as well as *efficient* – it had fast execution times. We also conducted preliminary experiments with Incremental NetProbe, which indicate that Incremental NetProbe results in significant speed-up of execution time with negligible loss of accuracy.

4.1 Performance on Synthetic Datasets

In this section, we describe the performance of NetProbe over synthetic graphs generated to be representative of real-world networks. Typical (non-fraudulent) interactions between people lead to graphs with certain expected properties, which can be captured via synthetic graph generation procedures. In our experiments, we used the Barabasi-Albert graph generation algorithm to model real-world networks of people. Additionally, we injected random sized bipartite cores into these graphs. These cores represent the manner in which fraudsters form their sub-networks within typical online networks. Thus, the overall graph is representative of fraudsters interspersed within networks of normal, honest people.

Accuracy of NetProbe. We ran NetProbe over synthetic graphs of varying sizes. and measured the accuracy of NetProbe in detecting bipartite cores via *precision* and *recall*. In our context, precision is the fraction of nodes labeled by NetProbe as fraudsters who belonged to a bipartite core, while recall is the fraction of nodes belonging to a bipartite core that were labeled by NetProbe as fraudsters. The results are are plotted in Figure 4.

In all cases, recall is very close to 1, which implies that NetProbe detects almost all bipartite cores. Precision is almost always above 0.9, which indicates that NetProbe generates very few false alarms. NetProbe thus robustly detects bipartite cores with high accuracy independent of the size of the graph.

Scalability of NetProbe.

There are two aspects to testing the scalability of NetProbe, (a) the time required for execution, and (b) the amount of memory consumed.

The running time of a single iteration of belief propagation grows linearly with the number of edges in the graph. Consequently, if the number of iterations required for convergence is reasonably small, the running time of the entire algorithm will be linear in the number of edges in the graph, and hence, the algorithm will be scalable to extremely large graphs.

To observe the trend in the growth of NetProbe's execution time, we generated synthetic graphs of varying sizes, and recorded the execution times of NetProbe for each graph. The results are shown in Figure 6. It can be observed that NetProbe's execution time grows almost linearly with the number of edges in the graph, which implies that NetProbe typically converges in a reasonable number of iterations.

The memory consumed by NetProbe also grows linearly with the number of edges in the graph. In Section 5.1, we explain in detail the efficient data structures that NetProbe uses to achieve this purpose. In short, a special adjacency list representation of the graph is sufficient for an efficient implementation (i.e., to perform each iteration of belief propagation in linear time.)

Both the time and space requirements of NetProbe are proportional to the number of edges in the graph, and therefore, NetProbe can be expected to scale to graphs of massive sizes.

4.2 Performance on the EBay Dataset

To evaluate the performance of NetProbe in a real-world setting, we conducted an experiment over real auction data collected from eBay. As mentioned before, eBay is the world's most popular auction site with over 200 million registered users, and is representative of other sites offering similar services. Our experiment indicates that NetProbe is highly efficient and effective at unearthing suspicious bipartite cores in massive real-world auction graphs.

Data Collection. We crawled the Web site of eBay to collect information about users and their transactions. Details of the crawler implementation are provided in Section 5.1. The data crawled lead to a graph with 66,130 nodes and 795,320 edges.

Efficiency. We ran NetProbe on a modest workstation, with a 3.00GHz Pentium 4 processor, 1 GB memory and 25 GB disk space. NetProbe converged in 17 iterations and took a total of 380 seconds ($\sim$ 6 minutes) to execute.

Effectiveness. Since our problem involves predicting which

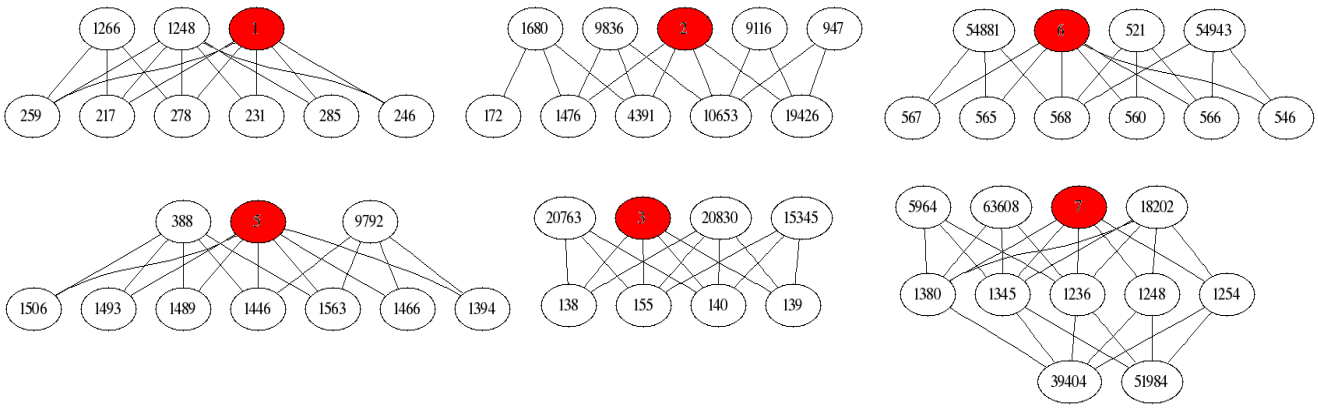

Figure 5: Cores detected by NetProbe in the eBay dataset. Nodes shaded in red denote confirmed fraudsters.

Fraud	Accomplice	Honest
0.0256	0.0084	0.016

Table 3: Fraction of negative feedback received by different categories of users

users are likely fraudsters, it is not easy to design a quantitative metric to measure effectiveness. A user who looks honest presently might in reality be a fraudster, and it is impossible to judge the *ground truth* correctly. Therefore, we relied on a subjective evaluation of NetProbe's effectiveness.

Through manual investigation (Web site browsing, newspaper reports, etc.) we located 10 users who were guaranteed fraudsters. NetProbe correctly labeled each of these users as fraudsters. Moreover, it also labeled the neighbors of these fraudsters appropriately so as to reveal hidden bipartite cores. Some of the detected cores are shown in Figure 5. Each core contains a *confirmed* fraudster represented by a node shaded with red color. This evidence heavily supports our hypothesis that fraudsters hide behind bipartite cores to carry out their fraudulent activities.

Since we could not manually verify the correctness of every fraudster detected by NetProbe, we performed the following heuristic evaluation. For each user, we calculated the fraction of his last 20 feedbacks on eBay which were negative. A fraudster who has already committed fraudulent activities should have a large number of recent negative feedbacks. The average bad feedback ratios for nodes labeled by NetProbe are shown in Table 3. Nodes labeled by NetProbe as fraud have a higher bad feedback ratio on average, indicating that NetProbe is reasonably accurate at detecting prevalent fraudsters. Note that this evaluation metric does not capture NetProbe's ability to detect users likely to commit frauds in the future via unearthing their bipartite core structured networks with other fraudsters.

Overall, NetProbe promises to be a very effective mechanism for unearthing hidden bipartite networks of fraudsters. A more exhaustive and objective evaluation of its effectiveness is required, with the accuracy of its labeling measured against a manual labeling of eBay users (e.g., by viewing their feedbacks and profiles, collaboration with eBay, etc.) Such an evaluation would be valuable future work.

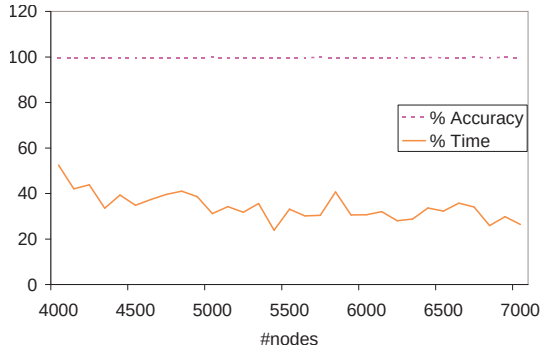

Figure 7: Performance of NetProbe over synthetic graphs with incremental edge additions

4.3 Performance of Incremental NetProbe

To evaluate the performance of Incremental NetProbe, we designed the following experiment. We generated synthetic graphs of varying sizes, and added edges incrementally to them. The value of h (see Sec 3.5) was chosen to be 2. At each step, we also carried out belief propagation over the entire graph and compared the ratio of the execution times and the accuracies with the incremental version.

The results are shown in Figure 7. Incremental NetProbe can be seen to be not only extremely accurate but also nearly twice as fast compared to stand-alone NetProbe. Observe that for larger graphs, the ratio of execution times favors Incremental NetProbe, since it touches an almost constant number of nodes, independent of the size of the graph. Therefore, in real-world auction sites, with graphs containing over a million nodes and edges, Incremental NetProbe can be expected to result in huge savings of computation, with negligible loss of accuracy.

5. THE NetProbe SYSTEM DESIGN

In this section, we describe the challenges faced while designing and implementing NetProbe. We also propose a user interface, which we believe is appropriate for visualizing the fraudulent networks detected by NetProbe.

Latest Feedback:	From	Date	Item#
excellent eBayer!! please visit us again	Seller island-ink (8349)	Apr-12-05	5763442208
Fast smooth transaction.	Seller discountwatersports (2058)	Jul-22-04	3818429146
excellant customer!	Seller sara-serval (287)	Jun-08-04	4303550234
Great communication.	Seller patan01 (152028) no longer a registered user	Jun-05-04	4191823868
Fast payment, great eBayer, A+++	Seller canarylady2000 (751)	May-20-04	4301759679
Very promp payment,	Seller crazysimon (8774)	Mar-11-04	3067601077
Super fast payment, A++++	Seller rusty05857 (651) no longer a registered user	Jan-16-04	3168839184

Figure 8: A sample eBay page listing the recent feedbacks for a user

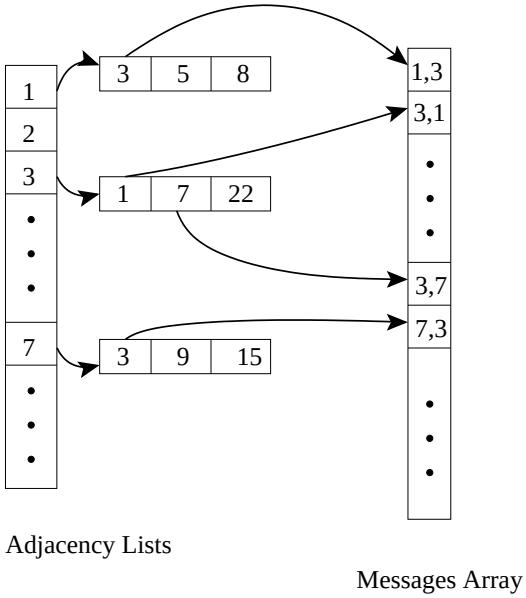

Figure 9: Data structures used by NetProbe's. The graph is stored as a set of adjacency lists, while messages are stored in a flat array indexed by edge identifiers. Note that the message sent from node i to j is always adjacent to the message sent from j to i.

5.1 Current (Third Party) Implementation

Currently, we have implemented NetProbe as a third party service, which need not be regulated by the auction site itself (since we do not have collaborations with any online auction site.) A critical challenge in such a setting is to crawl data about users and transactions from the auction site. In this section, we describe the implementation details of our crawler, as well as some non-trivial data structures used by NetProbe for space and time efficiency.

Crawler Implementation. EBay provides a listing of feedbacks received by a user, including details of the person who left the feedback, the date when feedback was left, and the item id involved in the corresponding transaction. A snapshot of such a page is shown in Figure 8. The username of each person leaving a feedback is hyperlinked to his own feedback listing, thus enabling us to construct the graph of transactions between these users by crawling these hyperlinks.

We crawled user data in a breadth-first fashion. A queue data structure was used to store the list of *pending* users which have been seen but not crawled. Initially, a seed set of ten users was inserted into the queue. Then at each step, the first entry of the queue was popped, all feedbacks for that user were crawled, and every user who had left a feedback (and was not yet seen) was enqueued. Once all his feedbacks were crawled, a user was marked as visited, and stored in a separate queue.

In order to crawl the data as quickly as possible, we enhanced the naive breadth-first strategy to make it parallelizable. The queue is stored at a central machine, called the *master*, while the crawling of Web pages is distributed across several machines, called the *agents*. Each agent requests the master for the next available user to crawl, and returns the crawled feedback data for this user to the master. The master maintains global consistency of the queue, and ensures that a user is crawled only once.

To ensure consistency and scalability of the queue data structure, we decided to use a MySQL database as the platform for the master. This architecture allows us to add new agents without suffering any downtime or configuration issues, while maintaining a proportional increase in performance. Further, each agent itself can open arbitrary number of HTTP connections, and run several different crawler threads. Thus, the crawler architecture allows for two tiers of parallelism — the master can control several agents in parallel, while each agent itself can utilize multiple threads for crawling.

The crawler was written in Java, and amounted to about 1000 lines of code. The master stored all of the data in a MySQL 5.0.24 database with the following schema:

```
User      (uid, username, date_joined, location,
           feedback_score, is_registered_user,
          is_crawled)
Feedback  (feedback_id, user_from, user_to, item,
           buyer, score, time)
Queue     (uid, time_added_to_queue)
```

We started the crawl on October 10, and stopped it on November 2. In this duration, we managed to collect 54,282,664 feedback entries, visiting a total of 11,716,588 users, 66,130 of which were completely crawled.

Data Structures for NetProbe. We implemented elaborate data structures and optimizations to ensure that NetProbe runs in time proportional to the number of edges in the graph.

NetProbe starts with graphical representation of users and transactions within them, and then at each iteration, passes messages as per the rules given in Equation 2. While edges are undirected, messages are always directed from a source node to a target node. Therefore, we treat an undirected edge as a pair of two directed edges pointing in opposite

directions. We use a simple adjacency list representation to store the graph in memory. Each (directed) edge is assigned a numeric identifier and the corresponding message is stored in an array indexed by this identifier (as shown in Figure 9).

Coming back to Equation 2, the second rule in this equation computes the belief of a node i in the graph by multiplying the messages that i receives from each of its neighbors. Executing this rule thus requires a simple enumeration of the neighbors of node i. The first rule however, is more complicated. It computes the message to be sent from node i to node j, by multiplying the messages that node i receives from all its neighbors except j. Naive implementation of this rule would enumerate over all the neighbors of i while computing the message from i to any of its neighbors, hence making the computation non-linear in the number of edges. However, if for each node i, the messages from *all* its neighbors are multiplied and stored beforehand (let us call this message as i's *token*), then for each neighbor j, the message to be sent from i to j can be obtained by dividing i's token by the last message sent from j to i. Thus, if the last message sent from j to i is easily accessible while sending a new message from i to j, the whole computation would end up being efficient.

In order to make this possible, we assign edge identifiers in a way such that each pair of directed edges corresponding to a single undirected edge in the original graph get consecutive edge identifiers. For example (as shown in Figure 9), if the graph contains an edge between nodes 1 and 3, and the edge directed from 1 to 3 is assigned the identifier 0 (i.e., the messages sent from 1 to 3 are stored at offset 0 in the messages array), then the edge directed from 3 to 1 will be assigned the identifier 1, and the messages sent from 3 to 1 will be stored at offset 1. As a result, when the message to be sent from node 1 to its neighbor 3 is to be computed, the last message sent from 3 to 1 can be quickly looked up.

NetProbe's fraud detection algorithm was implemented using these data structures in C++, with nearly 5000 lines of code.

5.2 User Interface

A critical component of a deployed fraud detection system would be its user interface, i.e., the "window" through which the user interacts with the underlying algorithms. For our scheme of detecting fraudsters via unearthing the suspicious network patterns they create, we propose a user interface based on visualization of the graph neighborhood for a user whose reputation is being queried. A screens-hot of the same is shown in Figure 10.

We believe that a simple and intuitive visualization tool is essential for users understand the results that the system produces. The detected bipartite cores, when shown visually, readily explain to the user why a certain person is being labeled as a fraudster, and also increase general awareness about the manner in which fraudsters operate. Users could finally combine the system's suggestions with their own judgment to assess the trustworthiness of an auction site user.

We have implemented the above interface to run as a Java applet in the user's browser. The user can simply input an username/email (whatever the auction site uses for authentication) into the applet and hit "Go". The tool then queries the system's backend and fetches a representation of the user's neighborhood (possibly containing bipartite core) in

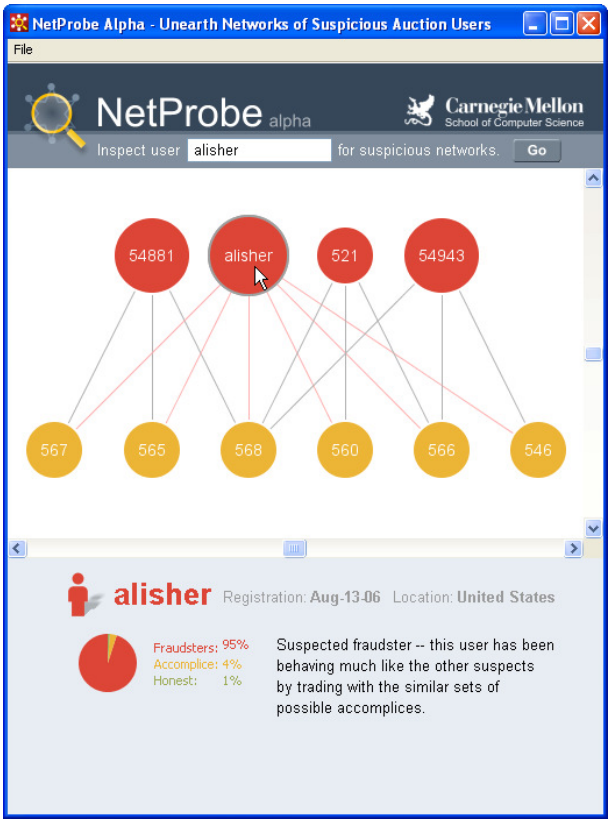

Figure 10: Proposed user interface for NetProbe

XML format. Such bipartite information could be pre-built so that a query from the user will most of the time lead to a simple download of an XML file, minimizing chances of real-time computation of the bipartite core information.

In summary, the user interface that we propose above provides a rich set of operations and visualizations at an interactive speed to the end user. We believe that displaying the characteristic patterns of a fraudster will be essential for the success of a fraud detection system.

6. SUMMARY AND FUTURE WORK

In this paper, we have described the design and implementation of NetProbe, the first system (to the best of our knowledge) to systematically tackle the problem of fraud detection in large scale online auction networks. We have unveiled an ingenious scheme used by fraudsters to hide themselves within online auction networks. Fraudsters make use of accomplices, who behave like honest users, except that they interact heavily with a small set of fraudsters in order to boost their reputation. Such interactions lead to the formation of near bipartite cores, one half of which consists of fraudsters, and the other is made up of accomplices. NetProbe detects fraudsters by using a belief propagation mechanism to discover these suspicious networks that fraudsters form. The key advantage of NetProbe is its ability to not only spot prevalent fraudsters, but also predict which users are likely to commit frauds in the future. Our main contributions are summarized in this section, along with directions for future work.

Neighbor state	Node state	
	Fraud	Honest
Fraud	$1-\epsilon$	ϵ
Honest	ϵ'	$1-\epsilon'$

Table 4: A sample instantiation of the propagation matrix for detecting cliques of fraudsters ($\epsilon' \ll \epsilon$)

Data Modeling and Algorithms. We have proposed a novel way to model users and transactions on an auction site as a Markov Random Field. We have also shown how to tune the well-known belief propagation algorithm so as to identify suspicious patterns such as bipartite cores. We have designed data structures and algorithms to make NetProbe scalable to large datasets. Lastly, we have also proposed a valuable incremental propagation algorithm to improve the performance of NetProbe in real-world settings.

Evaluation. We have performed extensive experiments on real and synthetic datasets to evaluate the efficiency and effectiveness of NetProbe. Our synthetic graphs contain as many as 7000 nodes and 30000 edges, while the real dataset is a graph of eBay users with approximately 66,000 nodes and 800,000 edges. Our experiments allow us to conclude the following:

- NetProbe detects fraudsters with very high accuracy
- NetProbe is scalable to extremely large datasets
- In real-world deployments, NetProbe can be run in an incremental fashion, with significant speed up in execution time and negligible loss of accuracy.

System Design. We have developed a prototype implementation of NetProbe, which is highly efficient and scalable in nature. In particular, the prototype includes a crawler designed to be highly parallelizable, while avoiding redundant crawling, and an implementation of the belief propagation algorithm with efficient graph data structures. We have also proposed a user-friendly interface for looking up the trustworthiness of a auction site user, based on visualization of the graph neighborhood of the user. The interface is designed to be simple to use, intuitive to understand and operate with interactive response times. The entire system was coded using nearly 6000 lines of Java/C++ code.

Directions for Future Work. From the algorithmic point of view, it would be interesting to see if NetProbe is able to spot patterns other than bipartite cores. Ideally, we should be able to formulate an instantiation of the propagation matrix to detect a desired patterns. For example, the matrix shown in Table 4 associates a high probability for a fraudster to connect to another fraudster, and should result in detection of cliques. Systematically analyzing the behavior of NetProbe for different patterns would be valuable future work. Another challenging problem is to learn the propagation matrix automatically from available data. Solving this problem entails development of appropriate machine learning techniques to use training data for learning both the form of the propagation matrix (i.e., the dimension of the matrix), and the value of all the entries.

7. REFERENCES

[1] Auctionbytes: ebay auction fraud spawns vigilantism trend. `http://www.auctionbytes.com/cab/abn/y02/m10/i12/s01`, 2002.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.

[3] D. H. Chau and C. Faloutsos. Fraud detection in electronic auction. In *European Web Mining Forum at ECML/PKDD*, 2005.

[4] D. H. Chau, S. Pandit, and C. Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. In *Proc. ECML/PKDD*, 2006.

[5] C. Chua and J. Wareham. Fighting internet auction fraud: An assessment and proposal. In *Computer*, volume 37 no. 10, pages 31–37, 2004.

[6] ebay inc. announces third quarter 2006 financial results. `http://biz.yahoo.com/bw/061018/20061018005916.html?.v=1`, October 2006.

[7] ebay: Avoiding fraud. `http://pages.ebay.com/securitycenter/avoiding_fraud.html`, 2006.

[8] Federal trade commission: Internet auctions: A guide for buyers and sellers. `http://www.ftc.gov/bcp/conline/pubs/online/auctions.htm`, 2004.

[9] Z. Gyongyi, H. G. Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB*, 2004.

[10] Internet fraud complaint center: Ic3 2004 internet fraud - crime report. `http://www.ifccfbi.gov/strategy/statistics.asp`, 2005.

[11] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[12] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. *VLDB*, pages 639–650, 1999.

[13] M. Melnik and J. Alm. Does a seller's ecommerce reputation matter? evidence from ebay auctions. *Journal of Industrial Economics*, 50:337–49, 2002.

[14] J. Neville and D. Jensen. Collective classification with relational dependency networks. In *2nd Multi-Relational Data Mining Workshop, SIGKDD*, 2003.

[15] J. Neville, . Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proc. SIGKDD*, 2005.

[16] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD*, pages 228–238, 2005.

[17] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM*, 43, 2000.

[18] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on ebay: A controlled experiment, 2003.

[19] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power-laws and the AS-level internet topology. *IEEE/ACM Transactions on Networking*, 11(4):514–524, 2003.

[20] Usa today: How to avoid online auction fraud. `http://www.usatoday.com/tech/columnist/2002/05/07/yaukey.htm`, 2002.

[21] W. Wang, C. Wang, Y. Zhu, B. Shi, J. Pei, X. Yan, and J. Han. Graphminer: a structural pattern-mining system for large disk-based graph databases and its applications. In *SIGMOD Conference*, pages 879–881, 2005.

[22] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.

[23] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD*, pages 324–333, 2005.

[24] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring AI in the new millennium*, pages 239–269, 2003.

[25] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD*, pages 797–802, 2006.