Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Broker-Assisting Trust and Reputation System Based on Artificial Neural Network

Bo Zong*†, Feng Xu*†, Jun Jiao*† and Jian Lv*†
*State Key Laboratory for Novel Software Technology,
Nanjing 210093, China
†Department of Computer Science and Technology,
Nanjing University, Nanjing 210093, China

***Abstract*—Due to the dynamic and anonymous nature of open environments, it is critically important for agents to identify trustful cooperators which work consistently as they claim. In the e-services and e-commerce communities, trust and reputation systems are applied broadly as one kind of decision support systems, and aim to cope with the consistency problems caused by uncertain trust relationships. However, challenges still exist: on the one hand, we require more flexible trust computation models to satisfy various personal requirements since agents in these communities are heterogeneous; on the other hand, trust and reputation systems calculate the trustworthiness of agents based on the agents' past behavior. The open environments are dynamic, agents are anonymous and the records about agents' past behavior are distributed in the environments, so agents have to search the required records through the environments due to their lack of valid information. Thus, efficient, scalable and effective information collection strategies are required to address these issues. In this paper we present a distributed trust and reputation system to cope with the challenges. We propose a novel and flexible trust computation model based on artificial neural networks. With the advantages of ANN, our trust model tunes the parameters automatically to adapt to various personal requirements. We propose a broker-assisting information collection strategy based on clustering method. With the support of brokers, subcommunities are managed by reputation mechanism in an efficient and scalable way and help their members collect information with high quality. We show the performance of our trust and reputation system by simulation.**

I. Introduction

In open environments, agents accomplish tasks and achieve goals by cooperating with each other(e.g. clients finish their tasks by using the services provided by servers). However, due to the dynamic and anonymous nature of open environments, agents have to cooperate with anonymous ones. Thus, it is critically important for agents to identify whether the cooperators are trustworthy and whether they work consistently as they claim in coming cooperations. *Trust and reputation system*(TRS) are proposed to identify trustful cooperators, since these problems caused by uncertain trust relationships cannot be solved by traditional security methods [10], [11].

In e-services and e-commerce communities, trust and reputation systems are applied broadly as one kind of *decision support systems*. Due to the success in web-applications(e.g. eBay [17], Amazon [18] and so on), the research on TRS has gained more attention in recent years. TRS is designed to help agents analyze trust relationships and make decisions before cooperations. Therefore the main tasks for TRS are quantifying trustworthiness between agents and helping agents make cooperation decisions. Meanwhile, the foundation of TRS is the cooperation history and cooperation records collected from agents. Consequently, to design a TRS we have to take two problems into consideration: 1). how to design a *trust computation model*(TCM) that provides more accurate trust value for decision procedure; 2). how to design an efficient and scalable *information collection strategy*(ICS).

A variety of TRS have been proposed in previous works [10], [11]. However, challenges still exist. On the one hand, agents' understandings on trustworthiness might be quite different, since agents in an open environment are heterogeneous. Though we try to adjust the parameters of a trust model manually to satisfy the personal requirements, without theoretical guarantee it is difficult to handle the adjustment procedure in practice. Thus, we require flexible trust computation models which can adjust the internal parameters automatically for different agents. On the other hand, trust and reputation systems depict the trustworthiness of agents by analyzing the transaction records. Without a central authority agents lack of valid information about other agents' past behavior, since the records are distributed in open environments, the open environments are dynamic and agents are anonymous. In order to improve the performance of TRS, an efficient, effective and scalable ICS is required, and it helps agents collect valid information and get rid of the noise information automatically.

In this paper, we propose a novel and flexible TCM based on *artificial neural networks*(ANN) to quantify the trust relationships between agents. Artificial neural networks are robust to noise data and support incremental training. Thus, our trust model tunes the parameters automatically to adapt to personal requirements taking the advantages of ANN. We propose a broker-assisting information collection strategy based on clustering method in order to improve the performance of the system. With the support of brokers, subcommunities are managed by reputation mechanism in an efficient and scalable way and provide information with high quality for their members.

The rest of this paper is organized as follows. In Section II

Supported by the National 973 Program under Grant No.2009CB320702, the National 863 Program under Grant No.2007AA01Z178 and No.2007AA01Z140, the NSFC under Grant No.60603034, No.60721002 and No.60736015 and the JSNSF under Grant No.BK2008017.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

we present an overview of current research on TRS. Section III introduces our proposed TRS. In Section IV we show the performance of our TRS by simulation. We give a conclusion and present our future work In Section V.

II. Related Work

Various trust and reputation systems have been proposed over the last decade. These models differ mainly in how they define trust and reputation, in how they quantify trust based on existing information, in their assumptions on how the systems obtains information from given networks, and in the set of trust attributes considered by them.

Abdul-Rahman and Hailes [1] proposed a model in which the trustworthiness falls into one of four discrete levels: *very trustworthy*, *trustworthy*, *untrustworthy* and *very untrustworthy*. Their work capture the most important characteristics of trust and propose the general structure for developing trust and reputation systems in open environments. Lik Mui et al. [2] described trust relationships by probability and proposed a computational trust and reputation model based on probability method. Sepandar Kamvar et al. [3] depict trust with real numbers and calculate the global trust between agents by iterated matrix computation. Our work integrates these previous works [1]–[3], and introduces artificial neural networks to improve the flexibility of trust computation models and to adapt to the personal requirements of different agents.

In previous works, referral-chain method [12], [13] is one of the most popular ways to collect information from agents. Xiong et al. [14] employ trust-set strategy to collect ratings from trusted agents. Nurit Gal-Oz et al. [4] use a clustering method to collect ratings from *similar* agents and prove that this strategy can protect agents from malicious attack effectively. Our work is based on previous works [4], [12]–[14] and introduces brokers to manage subcommunities by reputation mechanism in an efficient and scalable way.

III. Trust and Reputation System

In open environments, agents build two types of trust relationships: trust in cooperators and trust in recommenders. According to trustworthiness of cooperators, agents decides whether it will cooperate with them; taking trustworthiness of recommenders into account, agents decides how important the recommendations from other agents will be.

TRS is designed to help clients make beneficial decisions. In order to design a well-performing TRS, we have to take two problems into consideration: 1). how to design *trust computation model*; 2). how to design *information collection strategy*. However, agents evaluate other agents' behavior by different criteria since agents in open environments are heterogeneous [9]. Therefore the fixed global parameters for trust models might be inappropriate for some of agents in open environments. Though we can try to tune parameters manually to meet different personal requirements, this procedure is difficult to implement in practice due to the lack of theoretical guarantee. If trust models have the capability to tune the parameters automatically for various agents, the flexibility will be improved greatly. Moreover, when an agent requests transaction records from others, the records it received might be helpless and lead to confusion in decision procedures. These records are noise for the agent, even though the sources are honest agents. Different preferences, habits and backgrounds result in that the more information collected by trust models does not mean that trust models provide more accurate decision supports. In this paper, we assume that "less is more" [4]: while the information collected globally might be mixed by noise, the information collected from selected agents is more valid and the search is more efficient. With this assumption, each agent identifies those agents which have similar evaluation criteria and requests records from these similar ones instead of global search. To some extent, the agents which provide dishonest recommendations can be considered as the agents which have special evaluation criteria. Thus, without loss of generality we assume that agents are honest in recommendation but have different evaluation criteria.

In this section, we introduce our proposed trust and reputation system including the TCM based on ANN and ICS based on clustering method. First of all we will introduce basic definitions for concepts in this paper for convenience of following presentation.

A. Definition

In the trust and reputation community, the definitions of trust and reputation have never come to a common agreement. In this paper, we first give intuitive definitions for reputation and rating, and refer the definition of trust in [10], [11].

Definition 1 (Rating): The evaluation of a transaction between two agents, from various perspectives or general perspective, presented by either of them or both of them.

In the rest of this paper, we will use the term rating instead of transaction evaluation.

Definition 2 (Reputation): Reputation is a relatively objective evaluation about past behaviors of target agent and is derived from previous transaction ratings which could be direct experience or recommend information from others.

As mentioned above, agent builds two types of trust relationships: trust in cooperators and trust in recommenders. These two types of trust can be understood as *reliability trust*.

Definition 3 (Reliability Trust): The subjective probability by which an agent, A, expects that another agent, B, performs a given action on which its welfare depends.

For simplicity, we use the term trust instead of reliability trust.

As acknowledged by many researchers [5], [6], trust and reputation are context-related(e.g. a good reputation in tickets reservation can not infer a good reputation in schedule optimization). In this paper, without loss of generality we consider

trust and reputation under a uniform context(e.g. the tickets reservation services between clients and servers). Furthermore, ratings are provided symmetrically by both clients and servers, however, in this paper we only consider the ratings provided by clients for the convenience of discussion. In other words, we assume that only clients evaluate the trustworthiness of servers in the following.

B. Trust Computation Model

In open environments, the capabilities and the evaluation criteria of agents are not uniform [6]. For instance, some servers possess competitive services with high quality, while some clients possess less competitive services with relatively low quality. Nevertheless, both of them might meet the requirements of different groups of clients. From another perspective, some clients are very picky and only these service with high quality can satisfy their requirements, while some clients with relatively low requirements have more choices.

As acknowledged, it is a difficult problem to predict the character of a client. Furthermore, because the open environments are dynamic, it is much more complicated to predict the distribution of clients with different characters on specific time. It is necessary to improve the flexibility of trust models for the dynamic environments. Our proposed trust model based on *artificial neural networks* attempts to achieve the goal by learning examples to tune parameters automatically to meet various personal requirements.

1) Trust Model Based on ANN: Artificial neural network provides a general and practical method for learning vector-valued functions from examples and ANN learning is robust to noise in the training data [7]. ANN system is based on some basic units such as *sigmoids* [16]. A sigmoid can take vector inputs and calculates the result by nonlinear function. By post-processing, it produces an output as the input for other sigmoids or the final output. For instance, given vector inputs as x_1 through x_n, a sigmoid calculates the combination of vector, and then output a real number. We can represent the output $o(x_1, \ldots, x_n)$ computed by the sigmoid as :

$$net = \sum_{i=0}^{n} w_i x_i, \quad \textit{where } x_0 = 1,$$

$$o(x_1, \ldots, x_n) = \frac{1}{1 + e^{-net}},$$

where each w_i is a weight that determines the contribution of input x_i to the sigmoid output. The ANN is trained by backpropagation algorithm to approach the optimal parameters for fitting the training set of input-output pairs and the learning result is the tuned weights $w_0, w_1, \ldots, w_n$ for the sigmoid. A sigmoid is designed as Fig.1.

As noted, the representation power of one single sigmoid is limited. Multilayer networks learned by *backpropagation* are capable of expressing a rich variety of decision surfaces [16]. We use multilayer ANN to establish the trust models that quantify the trust relationships between agents.

Ratings are valid only within a period of time. To some extent, ratings implicate the capability of servers from clients' perspectives [4], [14]. If each rating contains two dimensions: time-stamp and evaluation, taking the most recent 20 ratings of target server into consideration, the relationship between the decision procedure and the rating for the transaction is depicted as:

$$(x_1, x_2, \ldots, x_{20}) \rightarrow eval.$$

For convenience, vector $(x_1, x_2, \ldots, x_{20})$ is represented as X, the relationship above is simplified as:

$$X \rightarrow eval,$$

where $eval$ is the rating and $eval$ can be real-valued(e.g. $eval \in [0, 1]$)or discrete-valued(e.g. $eval$ is either $satisfy$ or $dissatisfy$). Thus the input-output pairs $(X, eval)$ are the training instances for ANN.

Assuming that $eval$ is discrete-valued as $satisfy$ and $dissatisfy$, we can construct an ANN with 20 inputs and 2 outputs as showed in Fig.2.

While output o_1 represents the status of *satisfy* and output o_2 represents the status of *dissatisfy*. Given a trained ANN trust model, when client *p* requires the trustworthiness of server *q*, *p* first collects ratings about *q* from its rating pool or requests ratings from its *neighbors*. The significance of ratings from sources will be different. Obviously ratings collected by

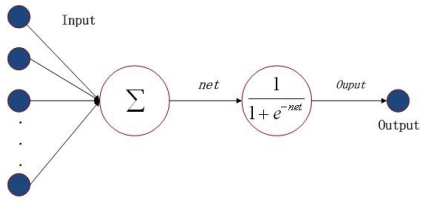

Fig. 1. The Sigmoid Unit

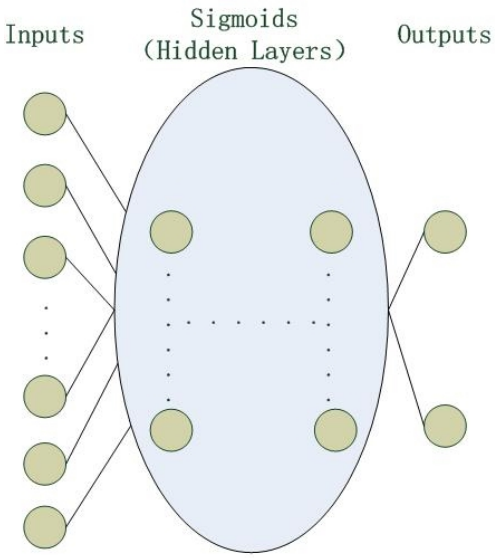

Fig. 2. General ANN Architecture

oneself are the most important. Without direct experience, the importance of ratings is hinged on the *similarity*, which we will introduce later, on characters of neighbors. Meanwhile, because the ratings are only valid during a period of time, we assume that a valid period consists of the most recent k time units. The most significant rating in each time unit is selected and the k ratings form an input for ANN trust model. After computation, the trust model provides two values o_1 and o_2 that we can consider as the confidence in *satisfy* and *dissatisfy* respectively. We can infer the trustworthiness $t_{p\to q}$ as:

$$t_{p\to q} = \frac{o_1}{o_1 + o_2},$$

2) Incremental Training: In open environments the amount of information is increased rapidly and it provides rich resources for ANN training. In order to satisfy personal requirements dynamically, it is important to make ANN trust model adjust itself to approach the optimal configuration. The optimal configuration for ANN is a set of parameters which are suitable enough to learn the underlying function. While an open environment is evolving, the potential optimal configuration might be changed subsequently. In this paper, we use the incremental training method to address this problem. Supposing that ANN is trained every *r* time units, with training data collected in last *r* time units, ANN is trained to learn current optimal configuration dynamically.

Besides, it is necessary to make ANN trust model adjust itself to approach the optimal architecture with the evolution of an open environment. The optimal architecture of ANN is a network large enough to learn the underlying function and as small as possible to generalize well [8]. A network smaller than the optimal one can not learn problems well while a network larger than the optimal one results in the lack of capability in generalization [7]. With existing incremental construction algorithm such as MOST [8], ANN is trained to improve its performance further. It is obvious that there is no need to search the optimal architecture all the time. When the performance for decision support is good enough and relatively stable, the search procedure is set on idle status. When notable decrease in performance is captured, the search procedure is set on active status and searches the optimal architecture based on current situation.

C. Information Collection Strategy

Given an established ANN trust model, when client *p* requires the trustworthiness of server *q*, *p* has to collect ratings first. The most significant ratings in each recent *k* time units are organized as an input for further decision support. It is acknowledged that ratings from oneself are the most significant, however, when lacking of direct experience, ratings from the neighbors, which have the highest *similarity* on character, are the most important.

In this paper, we propose an information collection strategy(ICS) based on clustering method. With the support of brokers, TRS maintains the property of subcommunities by reputation mechanism and quantifying the similarity between clients. Besides, this method has the nature that it can protect clients from malicious attack effectively by the maintenance algorithm [4]: malicious clients will be separated after long-term communication so that they cannot subvert the systems. We will use the existing clustering algorithms [15] to initialize the subcommunities. In the following, we will pay more attention on how to use brokers to maintain the property of subcommunities in an efficient and scalable way.

1) Broker: We envision that every subcommunity has its own broker, which might be implemented by a common and certified software package just like Microsoft® Passport or Liberty Alliance [6]. A broker is a delegate for a group of clients which have similar character, while there is at most one broker for single client. Brokers are in charge of managing subcommunities(e.g. a broker can grant the admission to competent applicants and deprive clients, which have been proved to be inappropriate for this subcommunity, of the member qualification).

Because most of clients are divided into subcommunities, a client, which does not belong to any subcommunity, is eager to enter an appropriate one where clients are willing to share experience with it and help it make more accurate decisions. Assuming that client *a* is willing to join subcommunity *c*, the application procedure is described as following steps:

1. Client *a* delivers an application to *b* which is the broker of *c*.
2. Broker *b* receives the application from *a*, *b* chooses k target servers which might be the most popular k servers in this community , might be the most infamous k servers or might be chosen randomly. Then *b* delivers a request for the recent ratings of these k target servers from *a* and ask *a* to give its response before time *t*.
3. Client *a* delivers the requested k ratings to *b*.
4. If broker *b* receives the response from *a* before the required time *t*, *b* chooses *m* target members in this subcommunity and delivers requests for similarity comparison; otherwise, *b* rejects the application.
5. Member *d* receives the request from *b*, computes the similarity $sim_{a,d}$ between *a* and *d* according to *predefined method*, and then sends the result to broker *b*.
6. Broker *b* collects results from requested *m* members, and take the combination of these results as final similarity sim_a for client *a*. If sim_a is greater than pre-defined threshold β, client *a* is invited into this subcommunity and has all the rights as member; otherwise, broker *b* will reject the application of *a*.

The discussion for similarity computation methods is beyond the scope of this paper, and here we provide an intuitive method. The similarity between client *a* and member *d* in the subcommunity is computed as:

$$sim_{a,d} = 1 - \frac{\sum_{i=1}^{k} |eval_{a,i} - eval_{d,i}|}{k},$$

where $eval_{x,y}$ represents the ratings from client *x* to the *y*-th server assigned by broker. And the final similarity sim_a is

averaged over all similarity reports as:

$$sim_a = \frac{\sum_{i=1}^{m} sim_{a,i}}{m}.$$

When a client has joined one subcommunity, it has more opportunities to gain helpful ratings from members and receives more helpful decision support. At the same time it also has the obligation to share experience with other members and provide other members helpful ratings. Once a member is considered as an incompetent one by broker, the broker will deprive it of the member qualification. In our work, we use a reputation mechanism to decide whether a client is appropriate for one subcommunity. When member *a* plans to initialize a transaction with server *s*, *a* will compute the trustworthiness of *s* by the ANN trust model. Due to the lack of direct experience, *a* will request ratings from other members. Member *a* first requests the *member-list* from broker *b*, and then sends out its requests to other members. The reputation mechanism begins to work here as below:

1. member *d* receives the request from *a*, and sends ratings back to *a*.
2. member *a* receives the ratings from *d* and continue to collects ratings needed until there are enough ratings to form an input instance.
3. By computation, *a* get a trustworthiness value $t_{a\to s}$ of server *s*. If $t_{a\to s}$ is greater than a pre-defined threshold θ, this transaction will be carried out and go to step 4; otherwise, transaction will be terminated.
4. After the transaction procedure, *a* will give a rating on the performance of *s*, and then *a* will report the input instance and final rating to broker *b*.
5. Broker *b* analyze the report, and then update the reputation of related members which have shared experience with *a* like *d*. Here, taking *d* as the example, if *b* find out that ratings provided by *d* make a positive contribution to this transaction, the reputation of *d* will increase; otherwise, the reputation value will decrease. If the reputation of *d* decreases and is below a pre-defined value ρ, member *d* will be considered as incompetent one in this subcommunity and will be deprived of the qualification.

Here we provide a simple reputation updating method as below:

$$r_d = \frac{p_d + C_1}{p_d + q_d + C_1 + C_2},$$

where r_d is the reputation of member *d*, p_d and q_d are the numbers of positive influence and negative influence caused by *d* respectively, and C_1 and C_2 are pre-defined constants. If $C_1 = 1$ and $C_2 = 1$, the initial reputation for a member is 0.5.

As presented above, the overhead for a client is the request for member-list from broker, the requests for ratings from other members and the final report of a transaction. This information collection strategy is of good efficiency and good scalability.

2) Broker-Assisting ICS: There are two types of clients: clients are in subcommunities and clients are not. Supposing that client *a* is in one subcommunity, when *a* collects information, this procedure is carried out as the steps presented above. Supposing that client *d* does not belong to any subcommunity, when *d* collects information, it requests ratings from its familiar clients. If a familiar client *e* is the member of one subcommunity, *e* will just provide part of its information(e.g. 50%) to *e*; otherwise, *e* can provide information as its will. For clients which do not belong to any subcommunities, the better performance on decision support motivates them to enter the appropriate subcommunities and this encourages clients to share transaction experience at the same time.

IV. Experiments

We evaluate our system in a simulation of e-services community in a peer-to-peer network developed on the PeerSim[1]. We implement the TCM and apply the implementation for the ANN provided by Weka[2].

A. Simulation Setup

For the sake of simplicity, all the agents focus on one type of services, and their evaluations for transactions are in the same context. Each agent in our simulation plays only one role, either client(services consumer) or server(services provider). It means that agents are divided into two groups: clients group and servers group. On the one hand, in the clients group, to a client out of any subcommunity, it only knows those agents directly connected with it and to a subcommunity member, with the help of broker, it can communicate with other members freely; on the other hand, in the servers group, servers do not communicate with each other for simplification. Besides, we assume that one client can communicate with any server at any time as long as the client wants.

Transactions between clients and servers are generated randomly. Before one transaction, the client organizes an input and the size of the input is 20(the latest 20 time units). Each client maintain a TCM, which is trained and works in the way above. The rating for transaction is combined with the input vector as a new instance for further incremental training. Additionally, the brokers generate the *test-list* randomly in our simulation.

Our simulation involves 20 different servers and 100 clients in which there are three types of evaluation criteria for the quality of services: *low*, *normal* and *high*, and we depict their characters with real numbers. The total number of simulation rounds is 1000. The period for incremental training is set as 100. Thus in the first 100 rounds we use the TCM based on *average method* to collect initial training data, and in the following rounds use the TCM based on ANN to continue the simulation. We set the parameters as: $\beta = 0.7$, $\theta = 0.5$ and $\rho = 0.6$. We run this simulation for 10 times and use the means of results as the evaluation criteria. After each transaction, a client will present a rating: either *satisfy* or *dissatisfy* according to its personal requirement.

[1]http://peersim.sourceforge.net/

[2]http://www.cs.waikato.ac.nz/ml/weka/

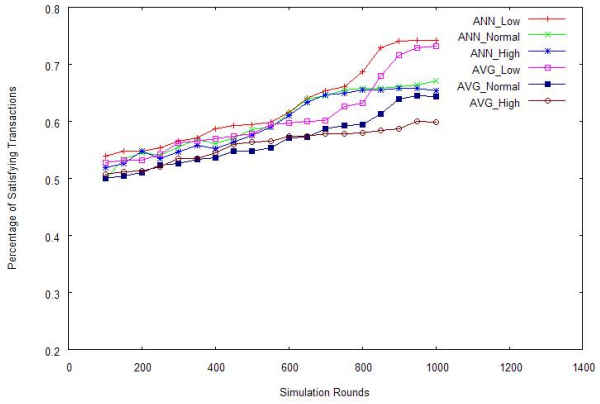

Fig. 3. Simulation Results

B. Results and Analysis

The goal of this simulation is to see whether the TCM based on ANN will help clients with different characters make accurate decisions. Thus we compare the performance, which is in terms of percentage of satisfying transaction, of a system consisting of clients with TCM based ANN and a system consisting of clients with TCM based on average method. Both systems employ the ICS proposed in this paper.

In Fig.3, we can see that the system using TCM based on ANN performs better than the system using average method on the *normal requirement* and *high requirement* clients, especially between Round 100 and Round 700. Because of the flexibility provided by ANN, the ANN system automatically adjusts the internal parameters to adapt to personal requirements and thus provides relatively more accurate decision support for different clients. However, after Round 700, we can see that performance of the AVG system begins to approach the performance of the ANN one. In our analysis, because both systems employ the same information collection strategy, when the subcommunities become stable and mature after 700 rounds, the ratings collected are more valid for decision support. Our ICS helps clients get rid of the noise information effectively and subsequently improves the performance of the AVG system.

In our simulation, we use simple ANN, which consists of three layers and only one hidden layer with three sigmoids. In the practical application, the environment might be much more complex and requires the use of self-adaptive construction algorithms for ANN.

V. Conclusion

In this paper, we propose a novel trust and reputation system with a flexible trust computation model and an efficient information collection strategy. we first propose a flexible trust computation model based on artificial neural networks which make trust models adapt to various personal requirements in open environments. In order to improve the performance of the system, we propose a broker-assisting information collection strategy based on clustering method. With the support of brokers, subcommunities are managed by reputation mechanism in an efficient and scalable way, and help their members collect information with high quality.

In future work, we will explore how to introduce the incremental neural network construction methods into trust and reputation system, which will further improve accuracy and flexibility of the system in real complex environments. We will design a more realistic simulation environment which has the ability to depict the characters of various agents in real complex environments. We think this work will help us verify the performance of our designed trust models more effectively. Additionally, taking multi-dimensional ratings into consideration, the complexity of personal requirements will be increased subsequently. We will make an attempt to design an ANN trust model which is flexible enough for multi-dimensional trust relationships.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In: Proceedings of the 33rd Annual Hawaii International Conference on System Science(HICSS, 2000).
[2] L. Mui, M. Mohtashemi and A. Halberstadt. A Computational Model of Trust and Reputation. In: Proceedings of the 35th Annual Hawaii International Conference on System Science(HICSS, 2002).
[3] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In: International World Wide Web Conference.
[4] N. Gal-Oz, E. Gudes and D. Hendler. A Robust and Knot-Aware Trust-Based Reputation Model. In: International Federation for Information Processing(IFIP), 2008, Volume 263; Trust Management II, (Boston: Springer), pp. 167-182.
[5] J. Hsu, K. Lin, T. Chang, C. Ho, H. Huang and W. Jih. Parameter Learning of Personalized Trust Models in Broker-Based Distributed Trust Management. In: Information Systems Frontiers 8(4): 321-333(2006).
[6] K. Lin, H. Lu, T. Yu and C. Tai. A Reputation and Trust Management Framework for Web Applications. In: IEEE International Conference on e-Technology, e-Commerce and e-Services(EEE 2005), Hong Kong, China.
[7] E. Bienenstock, S. Geman and R. Doursat. Neural Networks and Bias/Variance Dilemma. In: Neural Computation, Vol.4, pp. 1-58, 1992.
[8] O. Aran and E. Alpaydin. An Incremental Neural Network Construction Algorithm for Training Multilayer Perceptrons. In: Artificial Neural Networks and Neural Information Proceeding(ICANN/ICONIP'03), Istanbul, Turkey, 2003.
[9] Y. Wang and J. Vassileva. Bayesian Network Trust Model in Peer-to-Peer Networks. In: Agents and Peer-to-Peer Computing, Second International Workshop, AP2PC 2003, Melbourne, Australia.
[10] A. Jøsang. Trust and Reputation Systems. In: A. Aldini and R. Gorrieri(Eds.), Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures.
[11] A. Jøsang, R. Ismail and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. In: Decision Support System, 2005.
[12] D. Huynh, N. R. Jennings and R. Shadbolt. Developing an Integrated Trust and Reputation Model for Open Multi-Agent Systems. In: Proceedings of 7th International Workshop on Trust in Agent Societies, pages 62-77.
[13] M. G. Uddin, M. Zulkernine, and S. I. Ahamed. CAT: A Context-Aware Trust Model for Open and Dynamic Systems. In: Proceedings of the 2008 ACM Symposium on Applied Computing, Pages 2024-2029.
[14] Z.Xiong, Y. Yang, X. Zhang, D. Yu, and L. Liu. A Trust-Based Reputation System in Peer-to-Peer Grid. In: Online Communities and Social Computing, HCII 2007, LNCS 4564, pp. 228-235, 2007.
[15] J. Edachery, A. Sen, and F. Brandenburg. Graph clustering using distance-k cliques. In: Graph Drawing, pages 98-106, 1999.
[16] T. M. Mitchell. Artificial Neural Networks. In: Machine Learning, pages 81-127.
[17] http://www.ebay.com/
[18] http://www.amazon.com/