Service Graphs for Building Trust*

Pınar Yolum† Munindar P. Singh‡

† Department of Artificial Intelligence, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
pyolum@few.vu.nl

‡ Department of Computer Science, North Carolina State University
Raleigh, NC 27695-7535, USA
singh@ncsu.edu

Abstract. Information systems must establish trust to cooperate effectively in open environments. We are developing an agent-based approach for establishing trust, where information systems are modeled as agents that provide and consume services. Agents can help each other find trustworthy parties by providing referrals to those that they trust. We propose a graph-based representation of services for modeling the trustworthiness of agents. This representation captures natural relationships among service domains and provides a simple means to accommodate the accrual of trust placed in a given party. When interpreted as a lattice, it enables less important services (e.g., low-value transactions) to be used as gates to more important services (e.g., high-value transactions). We first show that, where applicable, this approach yields superior efficiency (needs fewer messages) and effectiveness (finds more providers) than a vector representation that does not capture the relationships between services. Next, we study trade-offs between various factors that affect the performance of this approach.

1 Introduction

We consider the problem of trust in large-scale, decentralized information systems that are represented by autonomous agents. In simple terms, the key problem is how an agent (or *trustor*) should trust another agent (or *trustee*). Trust is for a purpose. That is, a trustor would (or would not) trust a trustee for a particular service. For this reason and to relate our work to the recent interest on Web Services, we consider a setting wherein different agents consume and provide information services to one another [1]. The agents offer varying levels of trustworthiness to others and are potentially interested in finding trustworthy agents who provide the services that they need.

Trust can be established through three major means. *Institutional* trust or trust in authoritative institutions or organizations is common in the off-line world. People trust in the power of these institutions to stabilize their interactions [8, p. 26]. Current distributed trust management approaches can be thought of formalizing institutional trust,

* This research was supported by the National Science Foundation under grant ITR-0081742. An earlier version of this paper was presented at the AAMAS-03 Workshop on Trust, Fraud, Deception and Privacy.

because they assume that digital certificates issued by various certificate authorities lead to trust [4].

That is, these approaches usually assume that trust is established merely through a chain of endorsements beginning with some trusted authority. However, only the most trivial level of trust can be established through such a mechanism. For example, knowing that a web-site carries a digital certificate issued by another known site does not guarantee that the web-site will act in a trustworthy manner.

For this reason, multiagent approaches seek to create trust based on *local* or *social* evidence. Social trust is built through information from others. This information could be testimonies from individual witnesses regarding the trustee, or from a reputation agency. The context in which the ratings were given as well as the evaluation of the services could vary by episode as well as by the parties that give the ratings. The credentials of the information sources (witnesses or reputation agencies) are crucial for interpreting this second-hand information correctly. Unless the agents that give the ratings are established to be trustworthy, their aggregate ranking would not be sufficient to create trust. That is, in order to create trust through second-hand information, the trustworthiness of the information sources must be established as well [15, p. 74]. A powerful way of ensuring that the sources themselves are trustworthy is by accessing them through *referrals* [14]. Local trust means considering previous direct interactions with a trustee, which often are the most valuable in creating trust for the following reasons. One, since the trustor itself evaluates the interactions, the results are more reliable. Two, the context in which the trustworthiness of the provider is evaluated is explicit and relevant to the trustor.

Previous agent approaches for trust emphasize either its local or its social aspects. By contrast, we develop an approach that takes a strong stance for both aspects. In our approach, the agents track each other's trustworthiness locally and can give and receive referrals to others. This approach naturally accommodates the above conceptualizations of trust: social because the agents give and receive referrals to other agents, and local because the agents maintain rich representations of each other and can reason about them to determine their trustworthiness. Further, the agents evaluate each other's ability to give referrals. Lastly, although this approach does not require centralized authorities, it can help agents evaluate the trustworthiness of any such authorities as well.

This approach enables us to address two properties of trust that are not adequately addressed by current approaches. One, trust often builds up over interactions. That is, you might trust a stranger for a low-value transaction, but would only trust a known party for a high-value transaction. Two, trust often flows across service types. That is, you might assume that a party who is trustworthy in one kind of dealings will also be trustworthy in related kinds of dealings.

Our main contributions are as follows. One, we introduce a graph-based representation of services, and show how it enables us to address the above two properties of trust. Two, we evaluate our graph-based representation by comparing it to a vector representation used in previous work, which is itself more advanced than a simple scalar representation. Graph representation enables trust to be propagated across service types, whereas vector representation does not capture the relation between services at all. Our results establish that the additional expressiveness of the graph representation helps:

a graph-based representation enables trustworthy service providers to be found more effectively and efficiently. Three, we perform a sensitivity analysis of the graph-based representation to identify factors that affect its performance further.

The rest of this paper is organized as follows. Section 2 introduces a graph-based representation for agents to model services. Section 3 describes our referrals-based approach for trust and our experimental setup. Section 4 compares this graph-based representation with a vector representation in terms of efficiency and effectiveness. Section 5 discusses and experimentally evaluates factors related to the performance of our representation. Section 6 discusses the relevant literature and outlines some directions for further study.

2 Graph-Based Representation

We consider a setting with a fixed number of service types. Service providers offer one or more of these services. Some of these services may be related, i.e., being a good provider for one may imply being a good provider for another. Conversely, some services may be unrelated to each other.

One way of representing the set of services is through a vector space model, where each element in the vector corresponds to a different domain and the weight of the element denotes the fitness of the service for that domain [14]. This is similar to the vector descriptions of documents in information retrieval. The vector representation is simple and quite effective if the elements are independent, since a vector representation does not capture any relationships between vector elements.

The second way is to represent the services as a graph, whose nodes map to service types. The graph representation is more expressive in that it can capture relationships between service types that a vector representation cannot. For example, a service provider that has been found to be trustworthy for one type of service can be considered for another type of service based on how well the services relate.

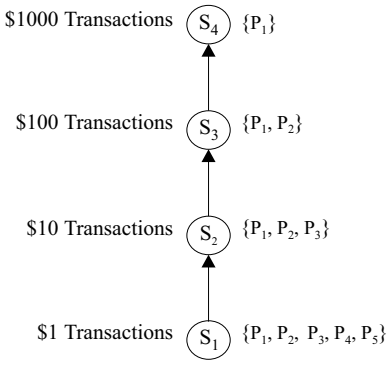

Fig. 1. A totally-ordered service graph

Figure 1 shows a simple graph. Here, each node represents transactions of different values. S_1 denotes transactions worth $\$1$, S_2 denotes transactions worth $\$10$, and so on. The list next to each node represents the trustworthy providers for that node. The agents trusted for a node are a subset of the agents trusted for the lower node. That is, if you trust someone for a $\$10$ transaction, you trust him for a $\$1$ transaction as well (e.g., P_3). The reverse need not hold. You might trust many for transactions of $\$1$ but probably only a few for $\$1000$ transactions (e.g., P_1).

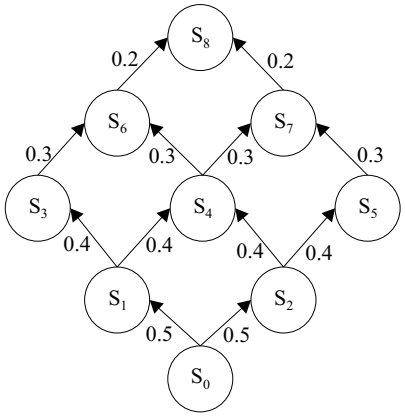

Fig. 2. An example service graph with weights

Figure 2 illustrates a setting with partially ordered services. Any two services that are related are joined by an edge. Here an edge $\langle s_i, s_j \rangle$ indicates that a provider who can perform s_i well may also be able to perform s_j well.

When an agent needs a provider for a service for which it knows of no providers, it can potentially ask others or *promote* a provider that it has used for another service. Promotions provide a systematic way to reuse previous experiences with the service providers. A provider is tried for a new service only if it has performed well for another service, and if performing well in the first service indicates that the provider may perform well for the second service. The likelihood of a service provider in a lower node to perform a service in the upper node is represented by weights on the edges. For example, the weight 0.5 from S_0 to S_1 means that a provider of S_0 will likely be providing S_1 half the time.

Notice that a service graph is maintained by each agent to autonomously capture its experiences. Thus agents may have differing weights for the same pair of services. The weights are adjusted independently by each agent. After delivering a service, a service provider is rated by the consumer. The rating reflects the satisfaction of the consumer. These ratings are used by the consumer to decide if this service provider will be used again or referred to other consumers. Service providers with low ratings are replaced with service providers that can potentially get higher ratings.

When promoting a provider from s_i to s_j, two factors are considered: how trustworthy the provider is for s_i and how related s_i and s_j are. We calculate the trustworthiness of the provider p at s_i (t_{pi}) through its ratings at s_i and the number of interactions (for s_i). The strength of the relation between s_i and s_j are given by the edge weight, w_{ij}.

$$(w_{ij} \times t_{pi}) > \theta \tag{1}$$

The product of the edge weight with the average ratings projects how much the agent can reproduce its ratings in s_j. If this projected value is greater than a promotion threshold θ, then the agent can be promoted to perform s_j.

Notice that in the extreme case, if $w_{ij} = 0$ (the services are not correlated), then the service provider is not expected to perform well in s_j even if it performs well in s_i. Conversely, if a provider is not trusted for s_i ($t_{pi} = 0$), then the provider will never be promoted to s_j irrespective of how correlated the two services are.

The weights that denote the relation between two services are estimated by each agent, which can update the weights in its graph based on its experiences. Hence, two agents can have different weights for the same edge. The graph weights are updated after promoting a provider and testing it for the higher service. The weights are tuned using a simple linear update mechanism. If a promotion from s_i to s_j is successful, i.e., if the provider gets a good rating in s_j as well, then w_{ij} is increased. Similarly, w_{ij} is decreased when a promoted provider gets a bad rating in s_j. The increase (or decrease) in the weight is proportional to the new rating of the service provider in s_j.

3 Experimental Setup

We investigate the properties of interest using agents who simulate requesting, providing, and evaluating services. The agents act in accordance with the following abstract protocol [17]. When an agent desires a service, it begins to look for a trustworthy provider for the specified service. The agent queries some other agents from among its *neighbors*, which are a small subset of the agent's acquaintances. A queried agent may either answer giving the identifier of a service provider who can potentially perform the desired service or may give referrals to other agents. The querying agent may accept a service offer, if any, and may pursue referrals, if any.

The agents are autonomous and may not respond to another agent. When an agent responds, there is no guarantee about the quality of the answer or the suitability of a referral. Likewise, no agent is necessarily trusted by others: an agent unilaterally decides how to rate another agent.

Each agent maintains models of its acquaintances, which describe their *expertise* (i.e., the quality of the answers they provide), and *sociability* (i.e., the quality of the referrals they provide). Each agent is initialized with the same model for each neighbor, but updates its models of its acquaintances based on interactions with them.

An agent that is generating a query follows Algorithm 1. Each agent starts by generating a query for a service (line 1). The distribution of requests for services captures the following intuition. In real life, we would expect most requests to be for services with intermediate risk rather than for services with little or too much risk. For this reason, we use a normal distribution to model the frequency of the incoming requests. As a result,

Algorithm 1 Find-Provider()

```
 1: Generate query for service type j
 2: promotedProviders = promoteLocally(j)
 3: if (promotedProviders != null) then
 4:    Add promotedProviders to providerSet
 5: else
 6:    Send query to matching neighbors
 7:    while (!timeout) do
 8:       Receive message
 9:       if (message.type == referral) then
10:          Send query to referred agent
11:          Record referral
12:       else
13:          Add answer to providerSet {answer contains a provider id.}
14:       end if
15:    end while
16: end if
17: for i = 1 to |providerSet| do
18:    Evaluate provider(i)
19:    Update agent models
20:    Update service graph
21: end for
```

the services S_0 and S_8 get the least number of requests, whereas services S_3, S_4, and S_5 get the most requests.

The agent promotes all the service providers that qualify to be promoted to perform this new service (line 2). If there are no such providers, then the agent sends the query to a subset of its neighbors (line 6). The main factor here is to determine which of its neighbors would be likely to answer the query. An agent that receives a query can either answer by returning the identifier of a service provider or giving a referral to another agent who is likely know of a service provider for the requested service.

If an agent receives a referral to another agent, it sends its query to the referred agent (line 10) and records the referral link (line 11). Simply put, the referrals generated for each query are used to update acquaintance models based on the quality of the service that is ultimately received from the providers found. After an agent receives a provider identifier or promotes a provider within, it evaluates the provider (line 18). We simulate this evaluation by looking up an evaluation value from a predefined table.

After the answers are evaluated, the agent uses its *learning policy* to update the models of its neighbors (line 19). In the default learning policy, when a good answer comes in, the modeled expertise of the answering agent and the sociability of the agents that helped locate the answerer (through referrals) are increased. Similarly, when a bad answer comes in, these values are decreased. Hence, the agents that give answers as well as the agents that give referrals are rated. At certain intervals during the simulation, each agent has a chance to choose new neighbors from among its acquaintances based on its *neighbor selection policy*. Key factors include the expertise and the sociability of the agents [18].

Algorithm 2 promoteLocally(j)

```
1: for i = 1 to |nodes| do
2:    for k = 1 to |providers(i)| do
3:       p = providers(i)(k)
4:       if (t_pi × w_ij > θ) then
5:          if (numberOfInteractions(p)≥ λ) then
6:             Add p to promotedProviders
7:          end if
8:       end if
9:    end for
10: end for
11: return promotedProviders
```

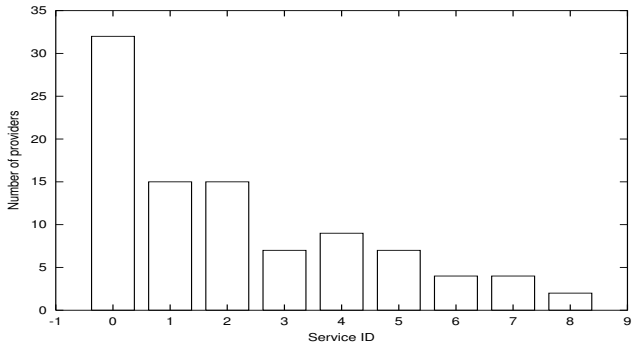

Fig. 3. Distributions of the service providers

The experiments use 100 service consumers and 32 service providers for nine types of services. Each agent has three randomly picked neighbors. Each agent generates 50 queries and may change its neighbors after every 5 queries. Each query denotes the desired service type; e.g., S_0, S_1, and so on. Notice that not all 32 service providers offer all the services. The key property we want to capture in modeling the distribution of the service providers is that in real life, we would expect more service providers to offer easier services than harder ones. Hence, the number of providers would decrease as the service gets more specialized. With this intuition, the experiments are set up such that most of the 32 service providers can perform services that are lower down the graph, whereas only a few of them can perform harder services, say, S_8, the most specialized service. We capture this intuition by decreasing the number of providers approximately by half between two consecutive nodes. For example, while 15 service providers offer service S_1, only 7 of them provide S_3. The number of service providers for each type of service is given in Figure 3.

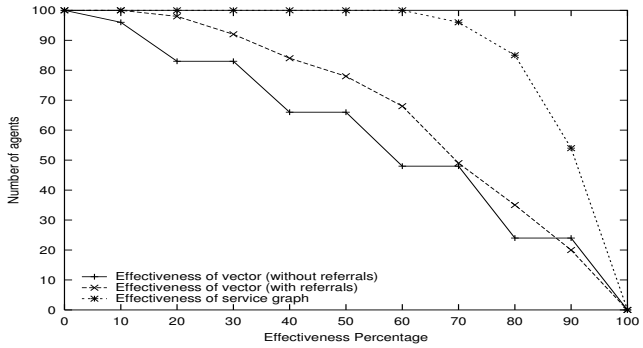

Fig. 4. The distribution of the agents for different values of effectiveness (After 10 queries)

4 Comparison of representations

Using this experimental setup, we compare the service graphs with the vector representation in terms of effectiveness and efficiency. A representation is effective if it allows agents to find the desired service providers. A representation is efficient if it allows the service providers to be found with as few messages as possible. In order to compare the effectiveness and the efficiency of the two approaches, the simulation is run with the same initial setup, same number of queries per consumer, and the same number of neighbors. After getting an answer, each consumer evaluates the service provider in the answer. The service providers that get a rating above a threshold are considered useful. For these experiments, the service providers provide services consistently. That is, a service provider that has provided a service will again perform the same quality of service.

To measure effectiveness, we find the percentage of the queries that have resulted in finding a useful service provider. That is, the ratio of queries that lead to useful service providers to all the generated queries is calculated. We look at the effectiveness after every five queries for the graph-based representation and the vector representation. We look at two cases of the vector approach: one with referrals, two without referrals.

Both in Figure 4 and in Figure 5, the x axis is the effectiveness percentage and the y axis is the number of agents. Both graphs plot the number of agents that achieve greater than or equal to the effectiveness percentage. The first graph shows the distributions after the 10th query and the second graph shows them after the 20th query.

In Figure 4, agents that employ service graphs achieve higher effectiveness than both of the vector approaches. The agents that use a vector with referrals generally do better than the agents without the referrals, except for one effectiveness value 90. That is, there are more agents that achieve at least 90 percent effectiveness in the vector approach without referrals, though the difference is minor.

The agents with the service graph achieve higher effectiveness rates in the second graph (Figure 5), too, though now the difference between the vector (with referrals) and the service graph approach is smaller. The performance of the vector approach increases as the agents learn about their neighbors and change their neighbors accord-

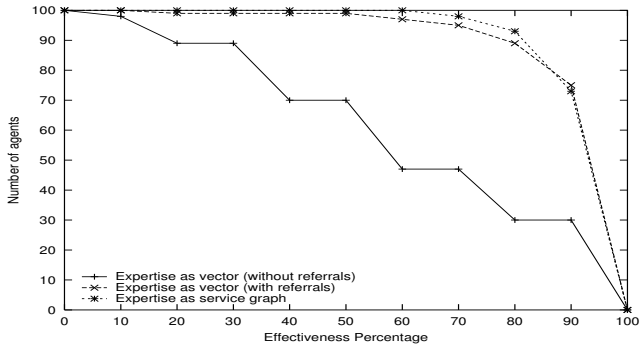

Fig. 5. The distribution of the agents for different values of effectiveness (After 20 queries)

ingly. After the 30th query, both approaches achieve an effectiveness rate of 99%, thus we do not show that in a different graph. However, when referrals are not employed, the effectiveness of the agents barely increases (Figures 4 and 5, solid lines). The average effectiveness for the no-referral case oscillates between 63% and 73%. Having no referrals causes two disadvantages to the agents. One, obviously they can pose their queries only to their neighbors, and incompetent neighbors cannot provide answers. Two, since there are no referrals, the agents interact with few other agents and learn only a small part of the society. Hence, when they change their neighbors, the set of agents they choose from is small and pseudo-random. Figure 6 plots the average effectiveness of all 100 agents after every five queries. We conclude that the consumers can locate trustworthy service providers more effectively with a graph-based representation than with a vector representation.

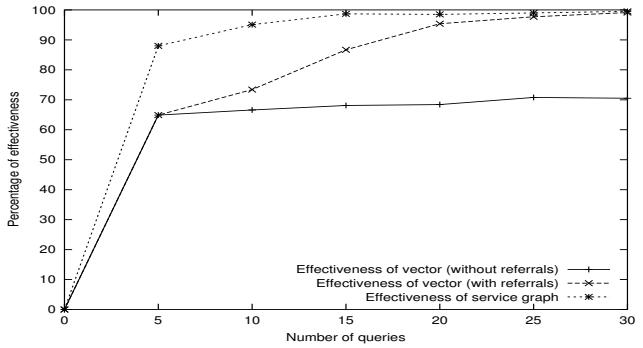

Fig. 6. Effectiveness of the representations

Next, we compare the average number of agents contacted per query (over 30 queries). Figure 7 plots this efficiency value for both approaches. The average value for the vector approach (with referrals) is 3.1, for the vector approach (without referrals)

2.81, and 0.45 for the service graph approach. In other words, the addition of referrals increases the number of contacted agents for the benefit of increased effectiveness. The service graph approach, on the other hand, yields a higher efficiency than both of the vector approaches as well as higher effectiveness.

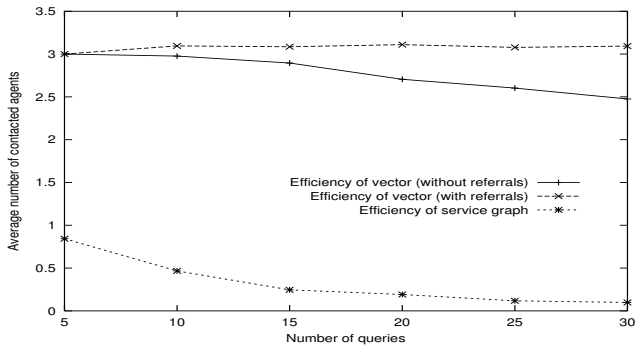

Fig. 7. Efficiency of the representations

Recall that initially, each agent knows of two service providers for possibly different services. For the results reported above, the initial distribution guaranteed that at least one agent in the system knows of a provider for each service. Consider a case, where none of the agents initially knew of a provider for S_8. In the vector approach, no matter how hard each agent searches for the provider through its neighbors, it will not be able to locate a provider for S_8. Whereas in the service graph approach, if an agent knows of a provider for S_6 or S_7, then it can promote the service provider to S_8. Thus, whereas the vector approach will definitely not find a provider, the service graph approach may find a provider through promotions from lower services.

Service graphs are most useful when the services are related, though even if the services are orthogonal the service graph would be equivalent to a vector representation. Thus, the service graph would in the worst case perform as well as the vector representation. There might only be one potential disadvantage. Following the previous scenario, assume that none of the service providers are trustworthy for service S_8. In this case, the service graph approach will promote service providers up only to find that they cannot fulfill S_8. Neither approaches will find a service provider, but the vector approach will use less time, whereas the service graph approach will try several providers (through promotions) and fail later.

5 Evaluation

We study how the initial setting, promotion threshold, and the number of previous interactions affect promotion accuracy and effectiveness of finding trustworthy service providers. For each experiment, we report averages from three simulation runs (additional runs yield similar results).

5.1 Control Variables

Initial Setting The initial environments can differ in two main ways. The first factor is how much the neighbors can help each other in finding service providers, since providers can be found through referrals as well as through promotions. To study the performance of the service graph representation, we seek to reduce the effect of referrals and prior knowledge of an agent. Therefore, we use a setting where each agent only knows of two providers for service S_0, the lowest service. This setting forces agents to promote the providers and test them for higher services. In addition, at least in the beginning, agents cannot give well-targeted referrals for higher services, since none of them knows of a trustworthy provider for higher services.

The second factor is related to how much the agents are initially willing to try new service providers. This factor, termed *trust prejudice*, captures whether an agent is willing to trust newcomers [6]. We capture this intuition through the initial graph weights. For example, if initially all the weights are 1, then the agents are willing to try out all new service providers in all types of services. Conversely, when the weights are all 0, the agents have the prejudice that no agents can be trusted.

We evaluate our approach using three initial settings. In the *homogeneous* setting, each agent starts with the graph shown in Figure 2. In the *trusting* setting, the graph edges are the same but the weights are higher (meaning the agents trust others more). In the *heterogeneous* setting, each agent starts with random weights on random edges of its own.

Promotion Threshold The estimated weight between two services is adjusted based on previous promotions between the two services. Intuitively, the promotion threshold denotes how much risk an agent is willing to take in its promotions. If the threshold for promoting up is low, then the agents will promote more providers, but might find out that more of these providers cannot perform the service. On the other hand, if the agents are reluctant to promote, then they might miss a chance to find a provider for a desired service. In Algorithm 2, θ refers to the promotion threshold.

Number of interactions The overall rating of a provider at the previous service should be reliable. It is widely accepted that the number of previous interactions increases the accuracy of the trust assessment [5]. That is, the average rating may not be representative if the total number of interactions are few. In other words, a service provider with a ranking of 0.7 over three interactions might be trusted more than a provider with a ranking of 0.8 over one interaction. In our approach, agents use the number of interactions as a gating factor so that only those providers that have proved sufficiently trustworthy in another service, which is sufficiently closely related to the service under consideration, and such that the agent has interacted with these providers often enough to trust them adequately. In Algorithm 2, λ refers to the required number of interactions.

5.2 Results

Promotion Accuracy Intuitively, high promotion accuracy captures the fact that only trustworthy service providers are promoted up the graph. Promotion errors are measured by the average number of wrong promotions performed by the agents.

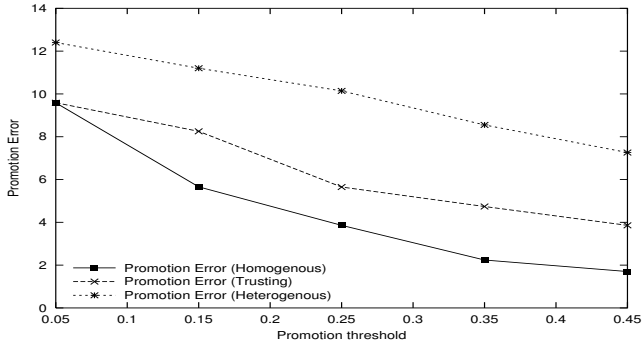

Fig. 8. Effect of initial setting

Figure 8 plots the promotion error for varying promotion thresholds. For all three curves, the error drops when the promotion threshold increases. That is, when agents take fewer risks, they make fewer mistakes. The heterogeneous setting has higher weights for more edges than the other two setups, and hence allows more promotions. For this reason, it is more prone to errors.

Next, we study the effect of number of interactions on promotion error. For each value of the promotion threshold, we plot the average promotion error. Figure 9 shows three plots for the homogeneous setting, corresponding to one, two, and three required interactions prior to promotion. The promotion error decreases with the number of previous interactions. For a threshold of 0.25, for example, when the required number of previous interactions is just one, the promotion error is almost 6. When the number of interactions is increased to two, the error drops below 4. When the number of interactions is further increased to three, the error becomes less than 2. In all three curves, increasing the promotion threshold decreases the promotion error, though the improvement is more significant for fewer interactions.

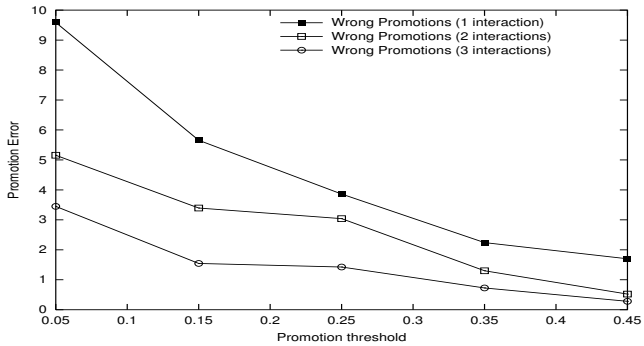

Fig. 9. Effect of previous interactions

Effectiveness Recall that effectiveness measures how often consumers find trustworthy providers for the desired services. Thus, achieving a high promotion accuracy is not enough for good performance. The agents should also achieve high effectiveness.

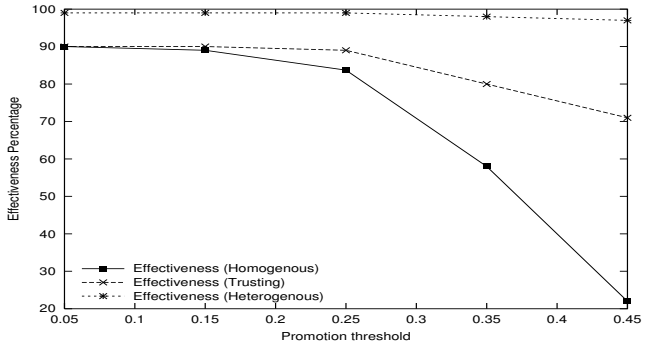

Fig. 10. Effect of initial setting

Again, we first look at the effect of initial setting on the effectiveness. Figure 10 plots three effectiveness curves for the three initial settings. This time the random setup achieves higher effectiveness than the other two setups. Since the random setup assigns weights to many edges, and hence allows more promotions, many providers—useful or not—are promoted and tested, resulting in almost always finding a provider.

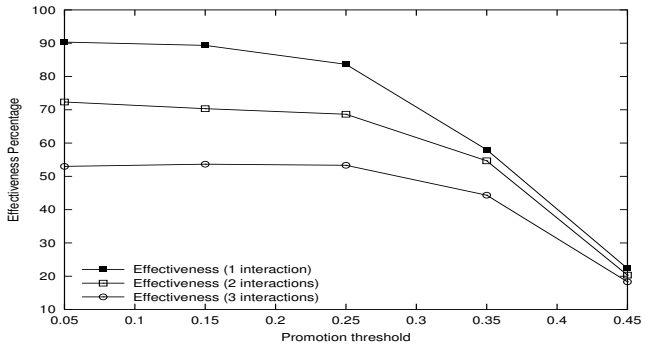

Fig. 11. Effect of the previous interactions on effectiveness

Figure 11 plots three effectiveness curves for varying values of the promotion threshold using homogeneous initial setting. Again, each curve corresponds to a case where different number of previous interactions is required. Independent of the number of interactions, if the threshold is high, the effectiveness is very low. Interestingly, for smaller

values of the threshold, we see agents achieve a higher level of effectiveness (find more trustworthy agents to interact with) if the number of interactions are fewer. This is the opposite of the curves for the promotion accuracy, where we saw that the number of interactions decrease the promotion error. In other words, high promotion accuracy rarely coexists with high effectiveness. For example, in Figure 9 when the number of previous interactions is set to three (with threshold 0.35), the promotion error is below 1. But, effectiveness for the same setup is not even 50%.

Performance = Effectiveness × Accuracy The reason for the inverse relation between promotion accuracy and the effectiveness is that if the consumers are cautious and promote reluctantly up the graph, they might miss many useful promotions, leading to sub-optimal effectiveness.

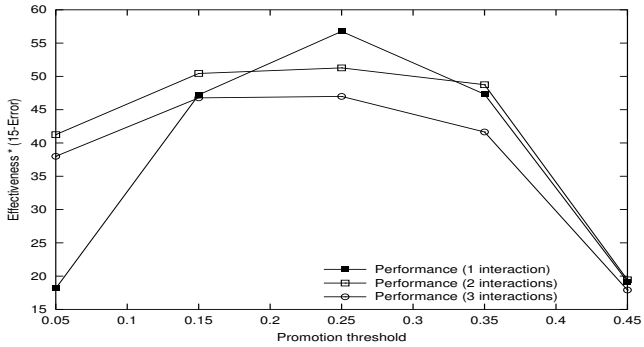

Fig. 12. Effectiveness and promotion error trade-off

Figure 12 plots this performance value based on Figure 9 and Figure 11. Neither extremes of the promotion threshold (0.05 and 0.45) achieve high performance. The lower threshold suffers from high promotion error, while the high thresholds lacks effectiveness. Optimal performance lies in the middle values of the promotion threshold. Among these, the performance is always better when the number of interactions is either 1 or 2. This suggests that the third interaction does not add much value to the performance. Among the 1 and 2 interaction cases, except for one value of the threshold (0.25), 2 interaction case outperforms the 1 interaction case. In general, this result suggests that it is better to be less cautious, trust more, and make some mistakes to be able to exploit a wider range of promotions.

6 Discussion

Lattice-based access control models have been used in computer security to regulate information flow [12]. Each node in the lattice denotes a different set of security privileges, called security classes. The more sensitive security classes are placed upper in the lattice. The information flow is only allowed from the lower security classes to the

higher ones. Thus, even though the less confidential information from lower security classes can be carried to the upper security classes, no confidential information flows down. This is similar to how we handle service types. Providers that can perform services higher up the lattice can also perform lower services. In addition, we promote providers from lower service types to higher ones based on the providers performance on the lower services.

Wille use concept lattices for knowledge discovery in databases [16]. The data objects are classified into meaningful concepts based on common attributes. The concepts, then are arranged in a line diagram, which represents the concepts and the subconcept relationships among concepts. This representation is a structured way to visualize and analyze information.

Referrals capture the manner in which people normally help each other find trustworthy parties [9]. MINDS, based on the documents used by each user, was an early agent-based referral system [3]. Kautz *et al.* model social networks statically as graphs and study some properties of these graphs, e.g., how the accuracy of a referral to a specified individual relates to the distance of the referrer from that individual [7].

Yu and Singh [19] develop an approach for distributed reputation management where a reputation of an agent is computed based on testimonies of the witnesses using the Dempster-Shafer theory of evidence. They show how this model can be used to detect agents that are non-cooperative or agents that abuse their reputation by slowly decreasing their level of cooperativeness. Since the witnesses are found through referrals, Yu and Singh's approach captures social trust. Local evaluations are captured through belief functions, but relationships among service types are not captured.

Barber and Kim [2] propose an approach wherein agents use a belief revision algorithm to combine evidence they receive from other agents. In addition to providing evidence, each agent specifies its level of confidence in the evidence. Barber and Kim's approach captures social trust, but contrary to our approach, the trustworthiness of agents who provide evidence are not considered. Their approach does not consider local evidence, i.e., the previous interactions of the trustor with the trustee.

Pujol *et al.* [10] develop an algorithm to find the reputation of an agent based on its position in a social network. The Web pages of users are taken as a basis to come up with the social network. If an agent is pointed to by agents with high reputation, then the agent is also considered to have high reputation, similar to the notion of authority exploited in search engines such as Google. Pujol *et al.* use their approach to find the reputations of authors where the reputation of an author is defined as the number of citations received. Even though each agent can calculate its own reputation based only on local information (i.e., the agents that point at it), a central server is needed to access others' reputations. This approach does not capture local trust, since direct interactions are not taken into account. It captures social trust since the reputation of an agent is derived through how other agents have linked to it, but has no means to correct it based on local observations of an agent. In other words, the link structure is static and the positions of the agents do not change based on their interactions. In our approach, we allow agents to change neighbors using the neighbors' ability to give referrals as a heuristic. This allows us to rate the sources.

Sabater and Sierra [11] develop a system for reputation management where reputations are derived based on direct interactions as well as the social relations of the agents. They use the number of interactions and the variance in ratings to derive the the trustworthiness of the agent through direct interactions. To assess the trustworthiness through indirect interactions, Sabater and Sierra use fuzzy inference to combine evidence from multiple witnesses. In this regard, their approach captures both social and local trust. On the other hand, Sabater and Sierra do not offer a mechanism to propagate trust across related services as we have done here.

Sen and Sajja [13] develop a reputation-based trust model used for selecting processor agents for processor tasks. Similar to our notion of service providers, each processor agent can offer varying performance. Agents are looking for trustworthy processor agents to interact with using only evidence from their peers. Sen and Sajja propose a probabilistic algorithm to find the number of agents to query to guarantee finding a trustworthy party. In our framework, we model the peers based on their prior performance and choose whom to ask for help based on these models. Thus, agents also decide the trustworthiness of the information source. However, in Sen and Sajja's framework, these models are not captured. All peers are treated the same independent of their previous behavior. This approach does not handle local trust, since previous interactions of an agent with processor agents are not taken into account.

The above approaches derive the trustworthiness of agents based on direct or indirect previous interactions. Our approach emphasizes the propagation of trust to related contexts as seen fit by an agent. In this respect, our graph-based representation complements the above approaches. Once the trustworthiness of an agent is derived, our approach can decide how this can be reused in other contexts.

7 Directions

Currently, we propagate trust based on a provider's trustworthiness for a single service. However, sometimes it would help to combine the trustworthiness of the provider in several services. For example, if a service is composed of several smaller services, the trustworthiness of the provider in all the subservices will affect the trustworthiness of the provider in the composed service. This problem is also acknowledged by Sabater and Sierra [11]. In future work, we plan to study such improvements to our model as well as evaluate our model with respect to different distributions for requesting and providing services.

References

1. A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. DAML-S: Semantic markup for Web services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, July 2001.
2. K. S. Barber and J. Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In R. Falcone, M. P. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, volume 2246 of *LNAI*, pages 73–82. Springer-Verlag, 2001.

3. R. Bonnell, M. Huhns, L. Stephens, and U. Mukhopadhyay. MINDS: Multiple intelligent node document servers. In *Proceedings of the 1st IEEE International Conference on Office Automation*, pages 125–136, 1984.
4. C. Castelfranchi and Y.-H. Tan. The role of trust and deception in virtual societies. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS 34)*, volume 7, pages 7011–7018. IEEE Computer Society Press, Jan. 2001.
5. R. Falcone and C. Castelfranchi. The socio-cognitive dynamics of trust: Does trust create trust? In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, LNAI 2246, pages 55–72. Springer-Verlag, 2001.
6. C. M. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW)*, LNAI 1647, pages 221–232. Springer Verlag, 1999.
7. H. Kautz, B. Selman, and M. Shah. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, Mar. 1997.
8. B. A. Misztal. *Trust in Modern Societies*. Blackwell Publishers, Cambridge, MA, 1996.
9. B. A. Nardi, S. Whittaker, and H. Schwarz. It's not what you know, it's who you know: Work in the information age. *First Monday*, 5(5), May 2000.
10. J. M. Pujol, R. Sangüesa, and J. Delgado. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 467–474. ACM Press, July 2002.
11. J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 475–482. ACM Press, July 2002.
12. R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, pages 9–19, Nov. 1993.
13. S. Sen and N. Sajja. Robustness of reputation-based trust: Boolean case. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 288–293. ACM Press, July 2002.
14. M. P. Singh, B. Yu, and M. Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, Apr. 2001.
15. P. Sztompka. *Trust: A Sociological Theory*. Cambridge University Press, UK, 1999.
16. R. Wille. Why can concept lattices support knowledge discovery in databases? In *Proceedings of the International Workshop on Lattice-based theory, methods and tools for Knowledge Discovery in Databases*, pages 7–20, 2001.
17. P. Yolum and M. P. Singh. Emergent properties of referral systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 592–599. ACM Press, July 2003.
18. P. Yolum and M. P. Singh. Self-organizing referral networks: A process view of trust and authority. In G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, volume 2977, pages 195–211. Springer Verlag, 2004.
19. B. Yu and M. P. Singh. An evidential model of distributed reputation management. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 294–301. ACM Press, July 2002.