Engineering Self-Organizing Referral Networks for Trustworthy Service Selection

Pınar Yolum and Munindar P. Singh, *Senior Member*

***Abstract*— Developing, maintaining, and disseminating trust in open, dynamic environments is crucial. We propose self-organizing referral networks as a means for establishing trust in such environments. A referral network consists of autonomous agents that model others in terms of their trustworthiness and disseminate information on others' trustworthiness. An agent may request a service from another; a requested agent may provide the requested service or give a referral to someone else. Possibly with its user's help, each agent can judge the quality of service obtained. Importantly the agents autonomously and adaptively decide with whom to interact and choose what referrals to issue, if any. The choices of the agents lead to the evolution of the referral network, whereby the agents move closer to those that they trust.**

This paper studies the guidelines for engineering self-organizing referral networks. To do so, it investigates properties of referral networks via simulation. By controlling the actions of the agents appropriately, different referral networks can be generated. This paper first shows how the exchange of referrals affects service selection. It identifies interesting network topologies and shows under which conditions these topologies emerge. Based on the link structure of the network, some agents can be identified as authorities. Finally, the paper shows how and when such authorities emerge. The observations of these simulations are then formulated into design recommendations that can be used to develop robust, self-organizing referral networks.

***Index Terms*— Multiagent Systems; Trust; Referrals**

I. Introduction

The Web is moving from a collection of pages to a collection of entities that provide and use *services*. Each service can involve tasks that vary from serving information such as Web pages to performing other complex tasks. The services are not merely distinguished by their domain or their tasks, but also in terms of other features of interest, such as the price, performance (e.g., throughput), or other domain-specific aspect. Hence, a service is described as aggregating multiple features.

The entities that provide and use services can be people or businesses, each potentially supported by an automated assistant. Because we deal with an open environment, the participating entities are autonomous and heterogeneous. Accordingly, we model them as agents in the computational system. The agents exercise their autonomy to decide the actions they perform, with whom they interact, or how they carry out their tasks.

The agents that *provide* the same service may differ in the way they implement their services, and the qualities of the services they provide. Each service provider can autonomously decide whom it serves and the quality of service it provides to each consumer. Likewise, the agents who use or *consume* the services vary in their needs and evaluations of services. Each service consumer can unilaterally set its own standards for the quality of service it would like to receive and potentially restrict its interactions to those that meet its standards.

In general, service implementations are not revealed to the consumers, nor to any other external parties. Not knowing the service implementation makes it hard to judge the quality of a service. Mechanisms based on a third-party evaluation of a service implementation cannot be employed. And, because the expectations of each consumer are different, a third party cannot evaluate service outcomes. Consequently, each consumer must evaluate the service it receives.

A. Trust.

Because the entities are autonomous and heterogeneous, selecting the right service providers is a significantly greater challenge in large scale open systems than in traditional distributed systems. The scale and dynamism of open environments imply that a participant would not know and would not be able to keep up with all potentially relevant participants. Large, open systems deviate from traditional systems primarily in the absence of central servers, even directory servers. The openness of such systems implies that there would be few regulatory restrictions for ensuring that the services offered are of a suitable quality; i.e., there are no guarantees about the quality of service provided by the participants. Hence, only those servers whose quality of service is acceptable by the participant will be relevant. Hence, it is crucial to locate useful participants and recognize them as *trustworthy*. This paper studies some key properties of trust.

For our purposes, trust is established between a service consumer and a service provider with respect to a particular service. Trust is inherently for a purpose and spans multiple dimensions. A service provider may be competent in some services but not in others. Accordingly, a consumer would (or would not) trust a provider for a particular service. For example, you may trust a travel agent for your travel needs but not for your medical needs. That is, trust is not a property of individual entities, but a property of relationships based on individual actions. To ascertain the trustworthiness of another

P. Yolum is with the Department of Computer Engineering, Boğaziçi University, 34342, Bebek, Istanbul, Turkey. E-mail: pyolum@cmpe.boun.edu.tr

M. P. Singh is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7535, USA. E-mail: singh@ncsu.edu

This research was supported by the National Science Foundation under grant ITR-0081742. This paper extends earlier versions that appeared in the AAMAS 2003 Conference [1] and AAMAS 2003 Workshop on Engineering Self-Organising Applications [2].

party, one must clearly formulate the service or individual actions in question. Even when these are made explicit, two consumers who interact with the same provider may have different assessments of the provider's trustworthiness. This variance could occur because of different evidence or different evaluations of the same evidence.

Accordingly, external third parties can neither establish the trustworthiness of others, nor dictate which individuals should be interacting with certain others. For this reason, each individual has to choose whom to interact with, judge whether they are trustworthy and establish trust in others based on her own means. Such a process for establishing trust is precisely based on the idea of *self-organization*. A self-organizing system consists of parties who act based on local interactions (without external control) and adapt to take into account useful parties [3], [4].

B. Referrals for Self-Organization.

A powerful way of ensuring that service providers and the information sources that recommend them are trustworthy is by accessing them through *referrals* [5], [6]. People commonly use referrals in real life to find useful providers; conversely, businesses use referrals from customers to find other potential customers. Referrals have been used in computational settings, but their usage has been restricted by rigid exchanges of the referrals, such as those used in domain name system (DNS).

We claim that flexible referrals are essential for locating trustworthy services, and we propose that referrals form a key organizing principle for engineering self-organizing applications that are targeted for open systems. Consumers can help each other find desired service providers by giving referrals to those trustworthy providers that have been useful for them. A consumer can judge the quality of the services received as well as the quality of the referrals (if any) that led to that provider. In other words, each consumer has an empirical basis for trust. More importantly, the consumers can self-organize by adapting to one another. For example, based on its interactions with others, a consumer can autonomously select the parties with whom to interact further. An agent would link to another party only if it has been useful in providing good services or in providing referrals that led to good services. Thus, the agents' associations with each other yield a *self-organizing referral network*.

C. Peer-to-Peer Systems

At an architectural level, consumers and providers of services are all *peers*, interacting without a need for a central server. Hence, our referrals-based architecture can be thought of as a peer-to-peer (P2P) architecture, where each peer acts as a client and a server, by requesting as well as serving information. Even though P2P architectures have been studied in the research community for many years, P2P applications have started to appear only recently. Gnutella [9] and Freenet [10] are two well-known P2P systems, geared for locating files. These systems allow peers to search for files in a network by propagating queries to other peers (i.e., without a centralized server) and to exchange files. Current studies of P2P systems focus on lower level properties of the systems, such as the naming schemes or bandwidth requirements. Our work emphasizes the higher level interactions. In other words, even when all architectural constraints are satisfied, participants still need to identify other useful participants with whom they can interact. For this reason, P2P systems must include an approach through which peers can help each other find trustworthy peers who offer high quality services. Even if some peers take on specialized functions similar to directory servers, others must establish that these specialized peers are indeed trustworthy, e.g., to ensure that their service recommendations are not based on ulterior motives, such as for paid-placement search engines, or that any ulterior motives are factored in to determine a suitable service.

D. Contributions and Organization

We have implemented a distributed platform using which adaptive referral systems for different applications can be built. To engineer and manage a referral system presupposes guidelines to adjust its behavior. This paper investigates some important guidelines for building such robust self-organizing referral systems. In order to identify such guidelines, it studies the behavior of self-organizing referral networks over simulations. The simulations give us the necessary controls to tune various policies and parameters. The framework and the test-bed that simulates the framework are powerful enough to capture many real-life details (see Section II-C).

This work first shows that referrals can induce a natural structure on the network of agents. This structure can then be used to identify different application domains. Using a particular application domain (i.e., e-commerce domain), the paper studies properties related to the performance and topology of referral networks. The performance properties study the efficiency and the effectiveness of referral networks. The analysis of these properties result in interesting guidelines for selectivity of referral exchanges.

The topology properties identify interesting network topologies. In certain contexts, some topologies can be harmful. We identify when this is the case. These topologies are important to point out because a network could be checked to see whether it is evolving into these pathological topologies, and if so, certain parameters can be adjusted to avoid them.

The rest of the paper is organized as follows. Section II describes our technical framework, including details of our model of referrals, the application domain, and our experimental setup. Section III studies effectiveness and efficiency of referral networks and identifies trade-offs between the two aspects. Section IV studies structural properties of referral networks, including undesirable network structures. Section V summarizes the main simulation results that can be incorporated as design guidelines to build referral systems. Section VI discusses the relevant literature and gives directions for further research.

II. Technical Framework

We consider a multiagent system whose members represent *principals* (people or businesses) providing and consuming

services. Services are understood abstractly, i.e., not limited to current Web services standards. Specifically, the services could involve serving static pages, processing queries, or carrying out e-commerce transactions, but their details are not represented in this paper. Our study will concentrate on self-organization of the agents.

The agents may offer diverse levels of trustworthiness and are interested in finding other trustworthy agents. An agent begins to look for a trustworthy provider for a desired service by querying some other agents from among its *neighbors*. The neighbors of an agent are a small subset of the agent's acquaintances, adaptively selected based on their usefulness.

The agents are autonomous. A queried agent may or may not respond to another agent by providing a service or a referral. The querying agent may accept a service offer, if any, and may pursue referrals, if any. When an agent accepts a service or follows a referral, there are no guarantees about the quality of the service or the suitability of a referral. We do not expect that any agent should necessarily be trusted by others: an agent decides how to rate another as it sees fit. Notice that trust applies both to the ultimate service provider and to the agents whose referrals lead to that provider.

Each agent maintains models of its acquaintances. Each model describes the *expertise* (the quality of the services it provides), and the *sociability* (the quality of the referrals it provides) of a given acquaintance. Both of these elements are adapted based on service ratings from the agent's principal. Using these models, an agent applies its *neighbor selection policy* to decide which of its acquaintances to keep as neighbors. Key factors include the quality of the service received from a given provider, and the value that can be placed on a series of referrals that led to that provider. In other words, the referring agents are rated as well.

The above framework accommodates the important properties of open and dynamic systems introduced in Section I. One, the agents can be heterogeneous. That is, agents can be of diverse designs and follow policies distinct from all others. Two, each agent operates autonomously based on its local policies. Three, each agent can adapt to the referral network by modifying its offerings and their quality, its policies, and its choice of neighbors.

A. Application Domains

The above framework enables us to represent different application domains. Two important domains are e-commerce and knowledge management, which differ in their notions of service and how the participants interact. In a typical e-commerce setting, the service providers are distinct from the service consumers. The service consumers lack the expertise in the services that they consume and their expertise does not get any better over time. However, the consumers are able to judge the quality of the services provided by others. For example, you might be a consumer for auto-repair services and never learn enough to provide such a service yourself, yet you would be competent to judge if an auto mechanic did his job well. Similarly, the consumers can generate difficult queries without having high expertise. For example, a consumer can request a complicated auto-repair service without having intimate knowledge of the domain.

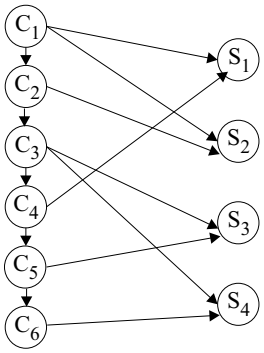

Fig. 1. A schematic configuration for e-commerce

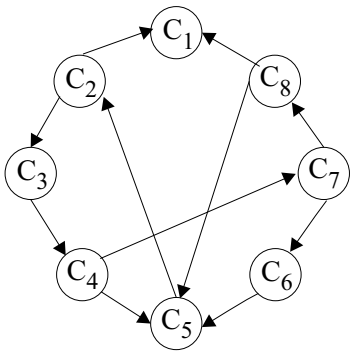

Fig. 2. A schematic configuration for knowledge management

Figure 1 shows an example configuration of service consumers and providers that corresponds to a commerce setting. The nodes labeled C denote consumers and the nodes labeled S denote service providers. The links between the node denote neighborhood relations. Consumers are connected to each other as well as to the service providers. These links form paths that lead to service providers. In this model, the service providers are dead ends: they do not have outgoing edges, because they neither initiate queries nor give referrals. Thus, their sociability stays low. Their concrete and modeled expertise may of course be high.

Figure 2 shows an example network configuration in a knowledge management setting. In this setting, the services are knowledge services, i.e., correspond to giving answers to queries. The consumers are not necessarily distinct from the service producers. An agent may be knowledgeable in one domain and hence respond to queries regarding that domain. Or, it might be looking for information services in another domain. Hence, all the nodes are labeled with C and denote consumers as well as producers. Each agent can generate and answer queries, as well as give referrals. This implies that potentially all agents can have nontrivial expertise and sociability. A consumer might lack the ability to evaluate the knowledge provided by someone who has greater expertise. However, agents would improve their knowledge by asking questions.

B. Agent Algorithms

Each consumer has varying levels of *interest* in receiving services. The interests and expertise of the agents are represented as term vectors from the vector space model (VSM) [7], each term corresponding to a different domain. The simulation uses these term vectors to generate queries and answers for the various agents.

Algorithm 1 Ask-Query()

```
1: Generate query
2: Compute a list of matching neighbors
3: Send query to matching neighbors
4: while (!timeout) do
5:    Receive message
6:    if (message.type == referral) then
7:       Send query to referred agent
8:    else
9:       Add answer to answerset
10:   end if
11: end while
12: for i = 1 to |answerset| do
13:   Evaluate answer(i)
14:   Update agent models
15: end for
```

Each agent is initialized with the same model for each neighbor. If the initial model of a neighbor corresponds to low expertise and low sociability values, the agent does not trust its neighbors enough to query them. For this reason, the initial neighbor model contains a high expertise value and a high sociability value. Thus, the initial model encourages the agents to query their neighbors.

An agent that is looking for an answer to a query follows Algorithm 1. An agent generates a query by slightly perturbing its interest vector, which denotes that the agent asks a question similar to its interests (line 1). More specifically, a simulation parameter p is set for the ratio of perturbation of the interest vector. The querying agent takes its interest vector and for each dimension i_k randomly generates a new value q_k from a range that is adjusted by the perturbation ratio p. Thus, for all dimensions of the query q_k, a random number is assigned between $(1-p) \times i_k$ and $(1+p) \times i_k$.

In applications that involve users, such as MARS [8], the first line of Algorithm 1 would correspond to a user request or an agent's anticipation of such a request. Next, the agent computes the list of neighbors that are likely to answer this query (line 2). We determine this through the *capability* metric. Then, the agent sends the query to the agents on the matching list (line 3).

The capability of an agent for a query measures how similar and how strong the expertise of the agent is for the query [6]. Capability resembles cosine similarity but also takes into account the magnitude of the expertise vector. What this means is that vectors with greater magnitude are regarded as indicating a higher capability for the given query vector. In Equation 1, Q ($\langle q_1 \dots q_n \rangle$) is a query vector, E ($\langle e_1 \dots e_n \rangle$) is an expertise vector and n is the number of dimensions these vectors have. This capability metric can also be used to measure how good an answer is for a given query.

$$Q \otimes E = \frac{\sum_{t=1}^{n} (q_t e_t)}{\sqrt{n \sum_{t=1}^{n} {q_t}^2}} \quad (1)$$

An agent that receives a query provides an answer only if its expertise matches the query. If it does, then the answer is the perturbed expertise vector of the agent. When an agent does not answer a question, it uses its *referral policy* to choose some of its neighbors to refer.

Back in Algorithm 1, an agent can receive messages from other agents. These messages can either be referral messages or answer messages. If an agent receives a referral to another agent, it sends its query to the referred agent (line 7). After an agent receives an answer, it evaluates the answer by computing how much the answer matches the query (line 13). If the answer matches the query more than a certain threshold, then the answer is considered good, otherwise bad. Since the answers are generated based on the expertise values of the agents, implicitly, the agents with high expertise end up giving the good answers. After the answers are evaluated, the agent updates the models of its neighbors (line 14). Whereas in the simulations the evaluations are performed with the capability metric, in real life applications, the agent would directly or indirectly evaluate the answer based on user feedback. 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. 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*. The number of neighbors is limited, so if an agent adds some neighbors it drops some neighbors as well. The underlying intuition is that an agent may interact with many other agents, but would only trust a small subset of these acquaintances.

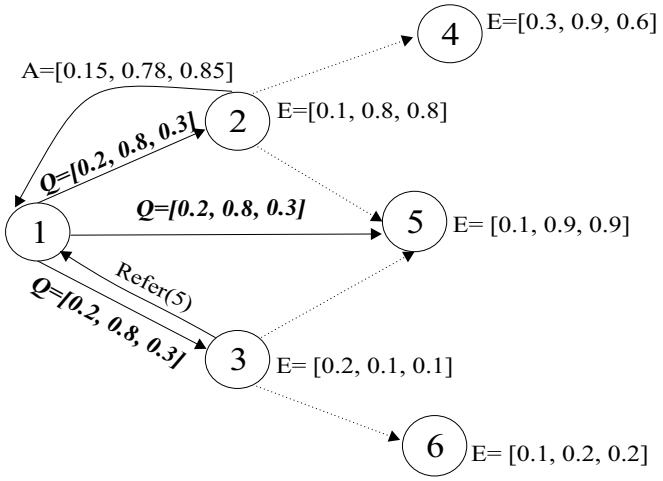

Fig. 3. An example search through referrals

Example 1: Figure 3 shows an example referral network, where the nodes denote agents. Agent 1's neighbors are agents 2 and 3, agent 2's neighbors are agents 4 and 5, and agent 3's

neighbors are agents 5 and 6. Agent 1 poses its query to its neighbors, agents 2 and 3. Agent 2 provides an answer, while agent 3 gives a referral to one of its own neighbors, agent 5. Agent 1 then sends its query to agent 5.

Together, the neighborhood edges among the agents induce the structure of the given society. In general, as described above, the structure is adapted through the decisions of the different agents. Although the decisions are autonomous, they are influenced by various policies.

C. Experimental Setup

This paper investigates the properties of the e-commerce domain via simulation. Studying the system through simulations enables us to study the mechanisms of the agent societies by giving us the necessary controls to adjust various policies and parameters. The findings of the simulation can be used to suggest certain kinds of mechanisms and representations for the agents themselves in real applications.

The simulations contain 400 agents, between 5% and 25% of which are service providers, and the remaining agents are consumers. The reported simulations contain interest and expertise vectors with 4 dimensions, where each dimension maps to one domain. Consumers have low expertise, since they do not offer any services themselves. The expertise of the providers and the interests of the consumers are distributed evenly over the domains. Each provider has expertise in just one domain whereas a consumer may have interests in multiple domains. The explained description of the population is fed into the simulator (Figure 4, Population Description Box).

Consumers have high *interest* in getting different services, but they have low expertise, since they do not offer services themselves. Providers have high expertise but low sociability. Since there are no humans to generate and evaluate queries, the interest vectors are used to generate queries and the expertise vectors are used to generate answers. Answers are evaluated using the capability metric on the query and answer vectors. A chain of referrals is followed for up to a given number of hops, and then dropped. Intuitively, longer chains make smaller contributions to trust. For the simulations reported here, the chain length is limited to 3.

Each agent has a fixed number of neighbors (4 to 8) and the same initial model for each acquaintance. In the beginning of the simulation runs, each agent is assigned neighbors randomly. During the course of the simulation, each agent interacts with other agents (i.e., acquaintances) and updates the models of its acquaintances (both expertise and sociability) based on the answers from the providers. After every two queries, agents can change their neighbors as they see fit. The simulations are run for 4 to 20 neighbor selections as specified below for each experiment. The details of the simulation, such as the number of neighbor selections, number of hops, and so on are provided to the simulator through a simulation description file as shown in Figure 4.

The simulation testbed is implemented in Java. Agents exchange messages using JBoss, a Java Message Service (JMS) implementation. The simulations reported here were performed on a PC with dual Pentium-3 500MHz processors and 1GB RAM, and running Linux. Each simulation was run with three different random seeds; averages of the three runs are reported.

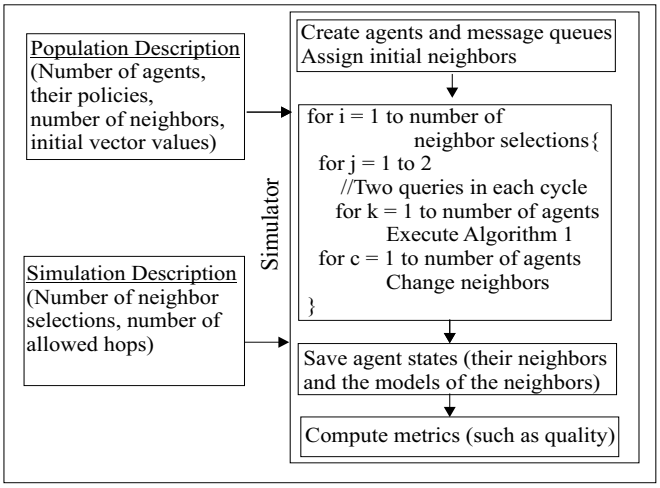

Fig. 4. A detailed architectural diagram

The simulator starts by creating agents and message queues based on the specifications in the the population descriptions. Next, each agent is randomly assigned neighbors. After this step, the simulator has a network of agents. The next step of the simulator is the main loop. This loop can be considered as the main simulation cycle and is repeated once for every neighbor selection. In each cycle, all agents generate two queries and follow Algorithm 1. At the end of the cycle, each agent considers its set of acquaintances and selects its set of neighbors using its neighbor selection policy. After the main loop of the simulation, the current state of the agents (their current neighbors and their models of acquaintances) are written to a file. Metrics necessary to analyze the network can be computed after the simulation ends.

III. Effectiveness and Efficiency

Effectiveness and efficiency of a referral network are key performance indicators. The effectiveness of a network measures how easily agents find useful providers. The efficiency of a network measures the ratio of good answers to number of agents contacted.

A. Effectiveness

We measure the effectiveness of the system using the direct quality metric and the n^{th} best quality metric. Both metrics are defined as obtained by an agent and then averaged over all agents.

The *direct quality* viewed by an agent reflects, via Equation (1), the usefulness of the neighbors of the agent, given its interest and their expertise. That is, it estimates the likelihood of the neighbors themselves providing good service.

Next, we take into account all other agents, not just the neighbors. Here, we measure how well the agent's interest matches the expertise of all other agents in the system, scaled down with the number of agents it has to pass to get to the

agent. That is, the farther away the good agents are from the given agent, the less is their contribution to the quality seen by the agent. Let I_i denote the interest vector of agent i and E_j denote the expertise vector of agent j. The contribution of agent j to agent i's quality is given by:

$$\frac{I_i \otimes E_j}{path(i,j)} \tag{2}$$

where the shortest path length is used in the denominator.

For a small population, it is reasonable to assume that each agent can potentially reach all other agents to which it is connected. But in a large population, an agent will be able to reach only a small fraction of the population. For this reason, instead of averaging over all agents, we take the n^{th} *best* measure. That is, we measure the quality obtained by an agent by its nth best connection in the network. The choice for n is nontrivial. If n is too big, each agent's quality would appear to be equally bad. However, if n is too small, the quality will reflect the neighbors quality as in the direct quality metric. For the results reported below, we use the nth best metric to measure an agent's quality and take n to be twice the number of neighbors that the given agent has.

A referral policy specifies which agents will be referred to a querying agent. We consider some important referral policies. We set the simulation variables appropriately so that an agent answers a query only when it is sure of the answer. This ensures that only the providers answer any questions, and the consumers generate referrals to find the providers.

1) *Refer all matching neighbors.* The referring agent calculates how capable each neighbor will be in answering the given query (based on the neighbor's modeled expertise). Only neighbors scoring above a given capability threshold are referred.
2) *Refer all neighbors.* Agents refer all of their neighbors. This is a special case of the matching policy with the capability threshold set extremely low (e.g., 0.1). This resembles Gnutella's search process where each node forwards an incoming query to all of its neighbors if it does not already have the requested file [9].
3) *Refer the best neighbor:* Refer the best matching neighbor. This is similar to Freenet's routing of request messages, where each Freenet client forwards the request to an agent that is the likeliest to have the requested information [10].

We study the effectiveness of different policies by varying the capability threshold. Figure 5 plots this threshold versus the quality of the network for different policies. In Figure 5, the lines marked *All Matching* show the *Refer all matching* policy for varying thresholds on the X axis. The case where the referral threshold is set to 0.1 denotes *Refer all.* The lines marked *Best Neighbor* plot *Refer best neighbor*, which is independent of the threshold.

Among the three policies, *Refer all* performs the worst for all three populations. As seen in Figure 5, when agents use this policy, the quality never exceeds 0.085. *Refer best neighbor* performs better than *Refer all matching* for small values of the capability threshold (e.g., 0.2). Compared to small thresholds, *Refer best neighbor* ensures a certain level of

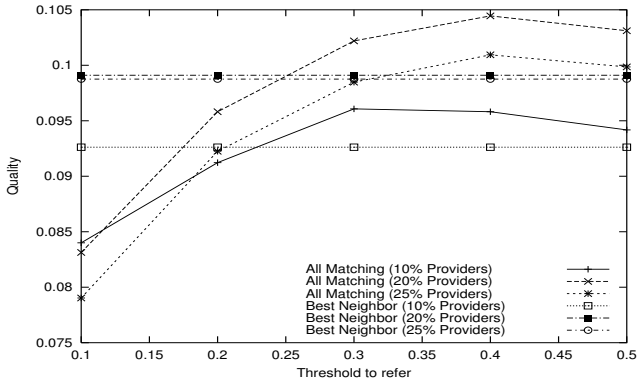

Fig. 5. Effectiveness of referral policies

selectiveness. Thus, it performs better than *Refer all matching* with small thresholds. For thresholds greater than 0.2, *Refer all matching* performs better than *Refer best neighbor*, where the best threshold increases with the percentage of providers in the society. *Refer all matching* with high thresholds are more selective than *Refer best neighbor.* Thus, higher thresholds generate better effectiveness than *Refer best neighbor.*

Observation 1: Exchanging more referrals does not guarantee that the quality of the network will be high. The topology of the network can prevent consumers from locating some of the service providers.

When agents are less selective in their referral policies, they exchange more referrals. However, sometimes even though referrals are exchanged, some agents may never be located, because they is no path to the provider from the requesting agent. When this is the case, exchanging more referrals does not help agents. A detailed analysis of this is presented in Section III-C.

B. Efficiency

Each agent in the referral network is autonomous and may well have different policies to take care of different operations such as answering a question or referring a neighbor. Thus, getting at a node closer to a target provider does not guarantee that the search is progressing. For example, in Figure 1 C_2 may ask C_3 but if C_3 is not responsive, then the search path becomes a dead-end. Hence, the quality metrics introduced above are optimistic; in actual usage, a provider may not respond and other agents may not produce helpful referrals. Hence, a high quality network does not necessarily mean that the agents will reach the services they are close to. To illustrate this point, we measure the efficiency of finding answers. Efficiency is defined as the ratio of the good answers received to the number of agents contacted. Figure 6 plots the capability threshold versus the efficiency for different referral policies.

Refer all matching with high thresholds (e.g., $0.4, 0.5$) yields the least efficiency. Since these policies are the most selective, few referrals are given. Hence, most of the time, the agents cannot find good answers, reducing the overall efficiency. However, an approximately equal number of good answers are found with both *Refer all* and *Refer all matching* with smaller thresholds, but because *Refer all matching* is more

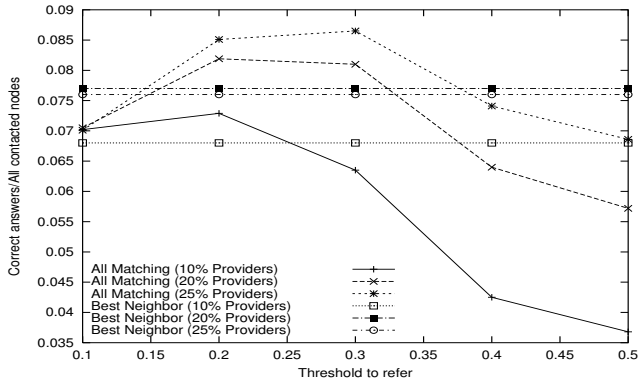

Fig. 6. Effect of selectivity on efficiency

selective, fewer referrals are generated, resulting in fewer agents being contacted. For this reason, *Refer all matching* with small thresholds produces higher efficiency than *Refer all*. *Refer best neighbor* is less selective than *Refer all matching* with high thresholds (e.g., 0.4, 0.5). With such high thresholds, *Refer all matching* may not yield a referral to any neighbor. However, with *Refer best neighbor*, one neighbor is always being referred to. Hence, *Refer best neighbor* is more efficient than *Refer all matching* with higher thresholds.

Observation 2: When few referrals are exchanged, good answers are not found. When more referrals are exchanged, good answers are found at the expense of contacting too many agents. Hence, it is better to be less selective in exchanging referrals to increase chances of finding good answers.

C. Analysis

We analyze the combined results on effectiveness and efficiency in three cases. First, with higher thresholds of *Refer all matching*, the agents can potentially reach the providers, but since referrals are given highly selectively, most of the time they cannot get referrals to locate the providers and pose their queries. Second, with smaller thresholds of *Refer all matching*, not only can the agents reach the providers, but since the referrals are less selective, they can locate the providers and get good answers. This is also the case for *Refer best neighbor*, although with this policy the number of good answers received is smaller. The third case is the most interesting one. With *Refer all*, agents get good answers although the quality of the network is poor.

When the agents exchange more referrals (using *Refer all* or for a lower threshold for *Refer all matching*), we would expect agents to be able to locate providers better and get closer to them. If the agents get good answers, then they are finding the providers, yet their ability to reach the providers (measured in quality) is still lower than with the other policies. The reason for this is that whereas the agents are close to a few providers (which ensures that they get good answers) they are isolated from many other useful providers. That is, the topology of the referral network may evolve in a way that isolates some of the providers from the consumers. The next section studies these possible undesirable topologies in greater depth.

IV. Network Topology

Recall that each agent chooses its neighbors based on local information only, without knowing which neighbors other agents are choosing. Even though each agent is doing the best for itself, the resulting graph may be undesirable.

At certain intervals during the simulation, each agent gets an opportunity to modify its selection of neighbors based on its acquaintance models. A *neighbor selection policy* governs how neighbors are added and dropped. Such policies can strongly influence the structure of the resulting graph.

What would happen if each agent chose the best service providers as neighbors? Or is it better to choose agents with higher sociability rather than higher expertise? To evaluate how the neighbor selection policies affect the structure, we compare three policies using which an agent selects the best m of its acquaintances to become its neighbors. Below, W denotes the weight assigned to sociability.

- *Weighted average.* Sort acquaintances in terms of a weighted average of sociability and how their expertise matches the agent's interests. (W is set between 0.1 and 0.9.)
- *Providers.* Sort acquaintances by how their expertise matches the agent's interests. (W is set between 0 and 0.1.)
- *Sociables.* Sort acquaintances in terms of sociability. (W is set between 0.9 and 1.)

The neighbor selection policies shape the topology of the network. That is, the network topology evolves differently based on how agents choose their neighbors. An obvious question is whether any one of these topologies are better than others or undesirable in certain settings. To answer these questions, we study well-known graph types from graph theory, namely bipartite graphs and graphs with weakly-connected components. We first study whether these topologies have any advantages or disadvantages over other topologies. Next, we study whether any one of the neighbor selection policies lead to such a topology.

A. Bipartite Graphs

A graph G is bipartite if it consists of two independent sets, i.e., two sets of pairwise nonadjacent vertices. When the simulation is started, we know that there is one independent set, the group of service providers. Since these do not have outgoing edges, no two service providers can have an edge between them. Thus the providers form an independent set. Now, if the consumers also form an independent set, then the graph will be bipartite. Essentially, the consumers' forming an independent set means that all the neighbors of all the consumers are service providers. Notice that if this is the case, then the consumers will not be able exchange referrals. If the graph becomes bipartite, the system loses all the power of referrals and all consumers begin operating solely on the basis of their local knowledge. For example, in Figure 7, the network that contains the nodes and the dotted and dashed lines form a bipartite graph. The consumers can reach the providers but not each other.

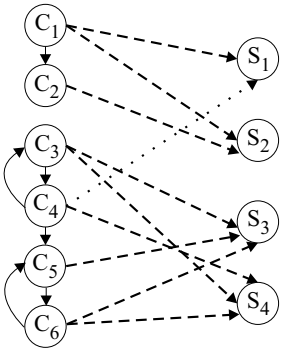

Fig. 7. Referral network configurations

Since the service providers do not have outgoing edges, they will not refer to any new agents. Thus, the consumers will not get to know new agents, and will not be able to change their neighbors, making the graph stable. However, for each agent there will be many agents that it cannot reach. Networks that allow reachability to these agents will have better quality and will thus be more desired than a bipartite graph. That is, the quality of the bipartite graph is not optimal. If the nodes of the network were rearranged into a topology other than that of a bipartite graph, the quality of the network could be higher.

Even if the graph is not bipartite, it could be extremely close to a bipartite graph. Let's say that the graph would be bipartite if a large subgraph is bipartite. In other words, removing a few edges from the graph would make the graph bipartite. This is still dangerous, since the graph might quickly evolve into a bipartite graph. Accordingly, we study the neighbor selection policies to see if they can cause the graph to turn into a bipartite graph. We use the number of edges needed to be removed as a metric for determining how close the graph is to becoming bipartite. We observe that when each agent exercises the *Providers* policy, if there are more providers than the number of neighbors an agent can have, then the graph converges to a bipartite graph. While this is the case for *Providers*, the same effect does not hold for *Weighted Average* or *Sociables*, since with these policies consumers may choose some other highly sociable consumers as neighbors.

Whereas detecting if a graph is bipartite is easy, determining the number of edges by which it differs from a bipartite graph is in general NP-complete [11]. Here, however, the semantics of the nodes serves to ease this problem. More specifically, one of the independent sets is already known, i.e., the set of providers. Let η denote the number of edges between the consumers. When η is smaller, the graph is closer to a bipartite graph. If there are no edges between consumers ($\eta = 0$), then the graph is bipartite.

Observation 3: When choosing neighbors, if agents prefer only expertise (i.e., use *Providers*), then the network can evolve into a bipartite graph, which prevents the consumers from exchanging referrals.

B. Weakly-Connected Components

A weakly-connected component of a graph is a maximal subgraph that would be connected when the edges are treated as undirected [12]. Thus, different components have disjoint vertices and are mutually disconnected. Consequently, consumers can at best find service providers in their own components. This means that if there is more than one weakly-connected component in a graph, then there is at least one consumer that will not be able to find at least one service provider. Consider again the network in Figure 7. If the network contains all the nodes and edges except the edge between C_4 and S_1 (shown with a dotted line), then the network will have two weakly-connected components. When that is the case, consumers C_1 and C_2 cannot locate service providers S_3 and S_4, since neither C_1 nor C_2 can receive referrals from the consumers that know S_3 or S_4.

We observe that in a population where each agent exercises *Sociables*, the graph ends up with more than one weakly-connected component. When agents follow *Sociables*, consumers link up with other consumers only since the consumers are the only sociable parties. This decreases the number of edges from consumers to providers. For example, again consider the same subset of the network in Figure 7, where the network contains all the nodes and edges except the edge between C_4 and S_1. After several iterations, if consumer C_5 were following *Sociables*, it could modify its choice of neighbors by linking to C_3 and removing its link to S_3. C_5 would do this because C_3 being a consumer would be more sociable than S_3, a service provider that does not provide any referrals. When all consumers act in accordance with *Sociables*, the providers could be totally isolated from the consumers.

Observation 4: When agents use *Sociables*, the network can become disconnected. This may prevent the consumers from locating some of the service providers.

C. Clustering

Watts defines the cliquishness of a graph as the likelihood of the neighbors of an agent being neighbors with each other [13]. The cliquishness coefficient for each agent i measures the ratio of actual edges among its neighbors to all the possible edges among the neighbors, as shown in Equation 3. Below, N_i denotes the set consisting of node i's neighbors. M_i denotes all the edges between the nodes in N_i.

$$\nu(i) = \frac{|M_i|}{|N_i|(|N_i| - 1)} \tag{3}$$

The cliquishness of a graph is then defined as the average $\nu(\cdot)$ of all the nodes in the graph.

Interest clustering denotes how similar the neighbors of an agent are in terms of their interests. Equation 4 captures the *similarity* between the interests of two agents, with the Euclidean distance between two interest vectors and normalizes it to get a result between 0 and 1. (I_i and I_j are of length n.)

$$I_i \oplus I_j = \frac{e^{-\|I_i - I_j\|^2} - e^{-n}}{1 - e^{-n}} \tag{4}$$

We measure interest clustering by a coefficient (Equation 5), similar in motivation to Watts' cliquishness coefficient. The interest clustering $\gamma(i)$ measures how similar the interest

vectors of an agent i's neighbors (including i itself) are to each other. The average of all the agents' interest clustering coefficients constitutes the interest clustering of the graph. $\gamma(i)$ is high if the neighbors of i are neighbors with each other and even higher if they have similar interests. In Equation 5, V_i denotes the set consisting of agent i and all of agent i's neighbors, and M_i denotes edges between the agents in V_i.

$$\gamma(i) = \frac{\sum_{(u,v)\in M_i} I_u \oplus I_v}{|V_i|(|V_i|-1)} \tag{5}$$

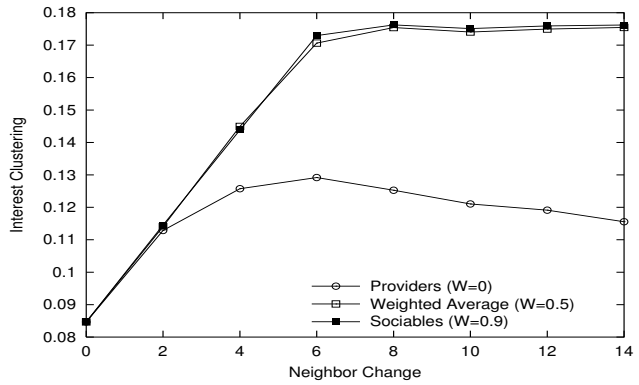

Fig. 8. Increase in interest clustering over neighbor changes

Figure 8 plots the interest clustering after every two neighbor changes for different neighbor selection policies. The interest clustering of the graph increases when the agents put greater emphasis on sociability when choosing neighbors.

Observation 5: When agents value sociability more (follow *Sociables*), agents with similar interests are more likely to become neighbors. The agents with similar interests may have located useful providers that match their own interests. These providers may also be useful for the given agent. Thus, the agents with similar interests can give well-targeted referrals and be considered sociable.

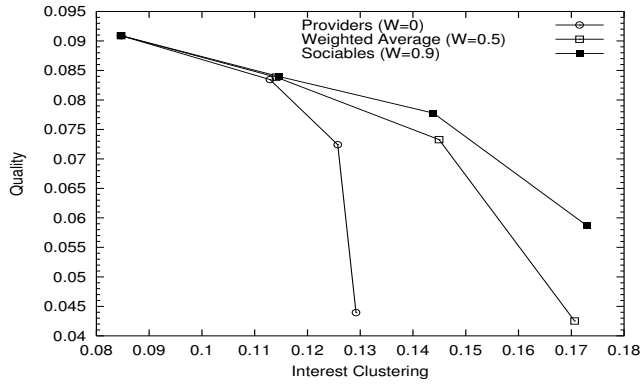

Fig. 9. Quality versus interest clustering

Next, we study the correlation between interest clustering and quality. Figure 9 plots the quality of the network for different values of interest clustering (after every second neighbor change). Each plot corresponds to a different neighbor selection policy.

We observe that interest clustering decreases with an increase in quality. A decrease in quality indicates that some consumers are getting farther away from the capable service providers. Meanwhile, if the interest clustering is increasing, then the agents are preferring to be neighbors with agents that are similar to themselves rather than with the service providers. Since the number of neighbors is limited, choosing agents with similar interests over those with high capabilities decreases the quality.

Observation 6: Becoming neighbors with agents with similar interests does not guarantee finding useful service providers.

D. Authoritativeness

The PageRank of a Web page measures its authoritativeness [14]. Informally, an authoritative Web page is one that is acknowledged to be highly accurate or reliable. A Web page has a high PageRank only if it is pointed to by Web pages with high PageRanks, i.e., if other authoritative pages view this page as authoritative.

Intuitively, the same metric can be applied to referral networks to measure the authoritativeness of agents. In the case of referral networks, an agent would be considered authoritative if it has been pointed to by other authoritative agents. Recall that an agent is pointed to by other agents if it is providing useful answers or referrals. Hence, if an authority finds another agent useful and points at it, then it is reasonable that this agent be considered an authority as well. That is, the agents decide on who is authoritative in the referral network.

The PageRank of an agent is calculated using Equation 6, where $P(i)$ denotes the PageRank of agent i, K_i denotes agents that have i as a neighbor, and N_j denotes the agents that are neighbors of j. In addition to accumulating PageRanks from incoming edges, each agent is assumed to get a minimum PageRank of $(1-d)$. Initially, each agent is assumed to be equally authoritative. Iterative computations of Equation 6 eventually stabilizes, yielding final authoritativeness values for each agent.

$$P(i) = d\sum_{j\in K_i}\frac{P(j)}{|N_j|} + (1-d) \tag{6}$$

For our calculations, we pick d to be 0.85 as is suggested in [14]; other values may also be reasonable. The calculated PageRanks are not normalized to demonstrate the variance in maximum PageRanks in different setups. The PageRanks of the agents are calculated centrally by building a graph from the neighborhood relations after the simulations. We study how the percentage of actual providers in the network and the policies that the agents follow affect the emergence of authorities.

1) Percentage of Providers: Intuitively, the percentage of agents with high expertise plays a crucial role in the distribution of PageRanks. For example, when there are too many service providers in the system, we expect that the PageRanks will tend to be shared among them. Having a small number of service providers may ensure that service providers with high authoritativeness will emerge. To study this point, we vary

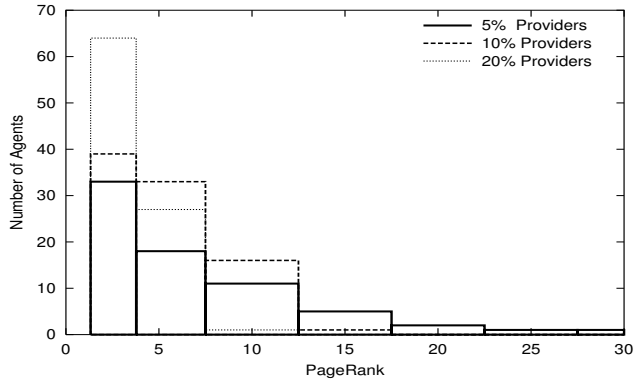

Fig. 10. PageRank distributions for percentage of providers

the percentage of the providers in the system. We study three populations with 5%, 10%, and 20% providers in them.

The histogram in Figure 10 depicts the PageRank distribution of three populations for PageRank values 2.5 and higher. The solid lines denote the population with 5% providers, the dashed lines denote the population with 10% percent providers, and the dotted lines denote the population with 20% providers.

Observation 7: When the percentage of providers is high, the PageRanks are clustered for small PageRank values. For example, when the population has 20% providers, the number of agents having PageRank higher than 2.5 is more than the cases for the other two populations. For the higher values of PageRank, the converse holds. For example, the only population that allows PageRanks higher than 25 is the 5% provider population. There is an implicit competition among the providers. When there are too many providers, they end up sharing the incoming edges. Therefore, only a few receive a relatively high PageRank. When there are a few providers, those providers tend to dominate more clearly.

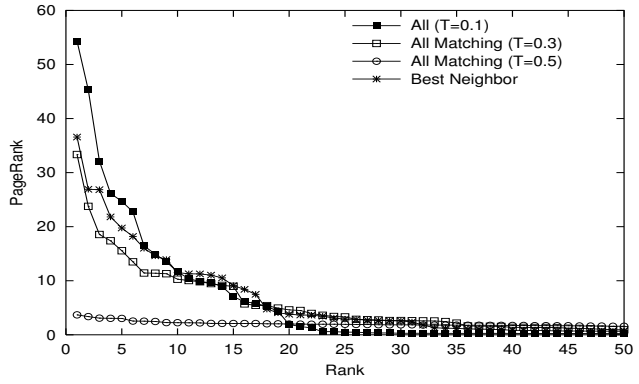

Fig. 11. PageRank distributions for referral policies

2) Referral Policies: Next we study the effect of referral policies in the emergence of authorities. Since the population with 5% percent providers allows the emergence of authorities more, we continue with this population. After each simulation run, the agents are ranked based on their PageRank. Figure 11 shows the PageRank distribution of the top 50 agents (out of a total of 400). If the agents use the *Refer all* policy, a few authorities with high PageRanks emerge. For example, the first 10 agents in the *Refer all* plot receive PageRanks greater than the first 10 agents in two instances of the *Refer all matching* plot (with thresholds 0.3 and 0.5).

Further, *Refer all* creates a large variance among the PageRanks. For example, whereas the first agent gets a PageRank of 54, the 50th agent gets a PageRank of only 0.23. Contrast this with *Refer all matching* with a threshold of 0.5, where the first agent gets a PageRank of 3.68 and the 50th agent gets a PageRank of 1.58. The distribution of PageRanks using *Refer best neighbor* falls between the distributions for *Refer all* and *Refer all matching* with high thresholds. In other words, when agents use *Refer best neighbor*, the highest PageRank is not as high as for *Refer all* (36) but the difference in PageRanks of the first and the 50th agents is still quite large.

Observation 8: Whereas more authorities emerge through *Refer all matching* (with different thresholds), *Refer all* causes the emergence of authorities whose level of authoritativeness is higher.

Intuitively, the explanation for the above is that *Refer all* is highly effective in disseminating information about the providers. Agents are thus more likely to encounter the providers and more likely to recognize their authoritativeness, thereby yielding high PageRanks for some of them.

3) Neighbor Selection Policies: Figure 12 plots the distribution of PageRanks with respect to some neighbor selection policies. The X axis shows PageRanks and the Y axis denotes the number of agents that get a PageRank greater than the PageRank shown on the X axis. W denotes the weight of the sociability in choosing a neighbor. The five plots correspond to *Providers* ($W = 0$), *Sociables* ($W = 1$), and three *Weighted average* neighbor selection policies with different weights.

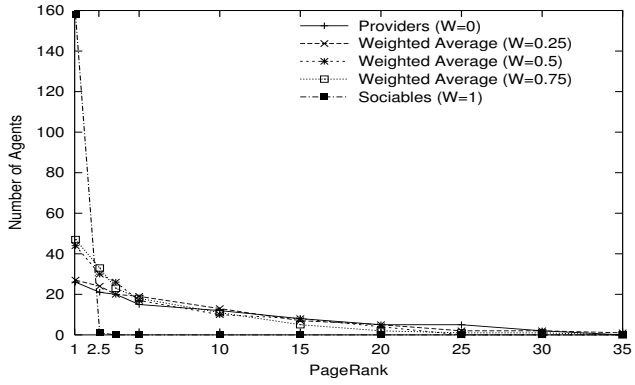

Fig. 12. PageRank distributions for neighbor selection policies

All curves, except the one for *Sociables*, are similar to each other. In all four cases, only a few authorities emerge. But, the level of their authoritativeness is high. For example, for *Providers*, while only 26 agents get a PageRank above 1, five of them get a PageRank above 20. Increasing the effect of the sociability slightly increases the number of agents with medium authority but slightly decreases the number of agents with high authority. For example, under *Weighted Average*, when the sociability and the expertise are weighted equally, the number of agents that get a PageRank above 1 is 44, while

four of them get a PageRank above 20.

Sociables does not follow this distribution. Initially, when not too many providers have been discovered, choosing neighbors only based on sociability does not help the agents find service providers. Hence, when agents follow *Sociables* in the beginning, most agents get average PageRanks (e.g., 158 agents get a PageRank around 1).

Observation 9: For strong authorities to emerge, it is important that the agents put a high value on the ability to produce high quality of service.

If the agents prefer sociables, there is little grounding in quality, and it is difficult to find good providers. Thus, strong authorities do not emerge. However, once the network has stabilized, sociability helps as there is a basis for good referrals, and thus there is value in those who can give good referrals.

V. Design Guidelines

Building applications of referral systems requires many design decisions. The above results yield design guidelines for real-life applications of referral systems. Here, we outline some possible applications of the properties observed in this paper.

Neither too many referrals nor too few referrals create high quality referral networks (Observations 1 and 2). Hence, in a referral system, it would be intuitive to encourage referrals but ensure that not an excessive number of referrals is exchanged. Similarly, some network topologies have been identified to be potentially undesirable in some settings such as e-commerce (Observations 3 and 4). A referral system could monitor if the network is evolving into these topologies and take further steps to prevent the network from exhibiting these properties. Observation 6 shows that becoming neighbors with similar interests does not guarantee quality. Accordingly, a referral system could also check whether agents are becoming clustered in small groups. When this is the case, the referral system can start functioning poorly.

As shown by Observations 7 and 8, having few providers in the system and exchanging referrals both help identify authorities. As shown in Observation 9, choosing neighbors only based on sociability discourages the emergence of authorities. Depending on the application, the emergence of strong authorities could be desired. Such an emergence shows that useful agents are identified and used by the others to find information. This is certainly important and could be enforced in a referral system. Even though the referral system cannot decide on the policies of its users, it can advise the users to adjust their policies appropriately. Conversely, for example, for a knowledge management domain, strong authorities would indicate that some agents answer substantially more queries than others. This overloading may not be desirable in such a setting. A referral system can then apply checks or use caching mechanisms to redistribute the expertise more evenly. When such caching mechanisms are used, agents can cache information that they have obtained from others and serve them as best suits them [15]. The identification of such properties of referral networks brings us closer to enabling self-organizing referral systems that can efficiently and effectively operate in open and dynamic environments.

VI. Discussion

We discuss some related approaches and point directions for further research.

Multiple Intelligent Node Document Servers (MINDS) was the earliest agent-based referral system [5]. Each node in the MINDS system is allocated a set of documents. Nodes help each other find documents in the network. Gradually, nodes learn how the documents are distributed in the network as well as the relevance preferences of individual users. Kautz *et al.* model social networks statically as graphs and study various aspects of their performance, such as the accuracy of the referrals, or the distance between a referrer and a questioner [16]. Our work, by contrast, seeks to uncover the structural properties of the network to design mechanisms that will improve the quality of the network.

Yu and Singh study referral networks in the context of scientific collaborations [17]. They show how the neighbor set size and referral graph depth affect locating agents accurately. Yu and Singh represent the referral process through weighted graphs, where weights are attached to both agents and referrals. They develop a method to minimize referral graphs so that agents only follow most promising referrals, i.e., referrals with high weights.

Kumar *et al.* develop an approach to infer web communities from the link structure of the Web [18]. Kumar *et al.* propose that any community structure should contain a bipartite core where the *fans* and *centers* make up the independent sets. Fans and centers are defined recursively, such that fans are pages that point at good centers and centers are pages that are pointed to by good fans. Kumar *et al.*'s approach assumes that if many fans point to the same set of centers, then they are likely to be on the same topic, and hence form a community. Our previous work on communities compared referral networks to that of Kumar *et al.* in depth [19].

Wang develops an approach for organizing agents into communities based on the similarity of their interests and expertise [20]. Initially, each agent registers with a middle agent randomly. Based on the queries received from the agents, the middle agents exchange agents to ensure that agents that have the same interests and expertise are handled by the same middle agent. This approach uses clustering to improve the efficiency of locating agents. When the agents' interests and expertise are more diverse, we believe that our Observation 6, i.e., clustering does not favor quality, will dominate.

Sabater and Sierra [21] develop a system for reputation management where reputations are derived based both on direct interactions and 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.

Buskens studies the effects of network structure on building trust [22]. Buskens simulates the interactions of buyers and

sellers that participate in iterated heterogeneous trust games that encourage the cooperation and discourage the cheating of participants. The buyers interact to exchange information on the trustworthiness of the sellers. The buyers themselves are assumed to be trustworthy. In our model, we do not assume that the consumers are trustworthy. Hence, we also take into account that some of the consumers may not give accurate referrals. Thus, each consumer also models the trustworthiness of other consumers. These models are then used to choose neighbors for future interactions. Buskens does not consider evolving network topologies, as we have done.

Shehory develops a decentralized approach for locating agents [23]. Rather than returning referrals as here, the neighbors themselves look for the desired agent. Shehory shows how increasing the average path length can increase the efficiency of agent location. In our approach, by choosing neighbors that are most suitable for itself, each agent increases its chance of getting good answers. By giving well-targeted referrals, each agent increases others' chances of finding good answers. These self-organizations aspects are not directly addressed in Shehory's approach.

Pujol *et al.* use the positions of agents in a social network to compute their reputation [24]. An agent receives a high reputation only if the agents that point to it also have high reputation, similar to the notion of authority exploited in PageRank. Pujol *et al.* calculate the reputations of authors where the reputation of an author is defined as the number of citations received. However, Pujol *et al.* do not study different network topologies as we have done here.

Our framework provides opportunities for further research. The above results here report the simulations performed in the e-commerce domain. One direction of research it to extend these results to other application domains, especially to knowledge management. Another direction of research is to incorporate other characteristics of applications, such as incentives for participation, into the framework.

REFERENCES

[1] 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)*, 2003, pp. 592–599.

[2] ——, "Self-organizing referral networks: A process view of trust and authority," in *Engineering Self-Organising Systems*, G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Eds., vol. 2977. Springer Verlag, 2004, pp. 195–211.

[3] G. D. M. Serugendo, N. Foukia, S. Hassas, A. Karageorgos, S. K. Mostéfaoui, O. F. Rana, M. Ulieru, P. Valckenaers, and C. van Aart, "Self-organisation: Paradigms and applications," in *Engineering Self-Organising Systems*, G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Eds., vol. 2977. Springer Verlag, 2004, pp. 1–19.

[4] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York: Touchstone, 2002.

[5] R. Bonnell, M. Huhns, L. Stephens, and U. Mukhopadhyay, "MINDS: Multiple intelligent node document servers," in *Proceedings of the IEEE International Conference on Office Automation*, 1984, pp. 125–136.

[6] M. P. Singh, B. Yu, and M. Venkatraman, "Community-based service location," *Communications of the ACM*, vol. 44, no. 4, pp. 49–54, 2001.

[7] G. Salton and M. J. McGill, *An Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

[8] B. Yu and M. P. Singh, "An agent-based approach to knowledge management," in *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 2002, pp. 642–644, poster.

[9] G. Kan, "Gnutella," in *[25]*, 2001, ch. 8, pp. 94–122.

[10] A. Langley, "Freenet," in *[25]*, 2001, ch. 9, pp. 123–132.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Comp., 1979.

[12] D. B. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

[13] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton: Princeton University Press, 1999.

[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[15] Y. B. Udupi, P. Yolum, and M. P. Singh, "Trustworthy service caching: Cooperative search in P2P information systems," in *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS)*, P. Giorgini, B. Henderson-Sellers, and M. Winikoff, Eds., vol. 3030. Springer Verlag, 2004.

[16] H. Kautz, B. Selman, and M. Shah, "ReferralWeb: Combining social networks and collaborative filtering," *Communications of the ACM*, vol. 40, no. 3, pp. 63–65, Mar. 1997.

[17] B. Yu and M. P. Singh, "Searching social networks," in *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2003, pp. 65–72.

[18] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Extracting large-scale knowledge bases from the Web," in *Proceedings of the 25th Very Large Databases Conference*, 1999, pp. 639–650.

[19] P. Yolum and M. P. Singh, "Dynamic communities in referral networks," *Web Intelligence and Agent Systems*, vol. 1, no. 2, pp. 105–116, 2003.

[20] F. Wang, "Self-organising communities formed by middle agents," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2002, pp. 1333–1339.

[21] 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)*. ACM Press, July 2002, pp. 475–482.

[22] V. Buskens, "The social structure of trust," *Social Networks*, vol. 20, no. 3, pp. 265–289, 1998.

[23] O. Shehory, "A scalable agent location mechanism," in *Intelligent Agents VI: Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1999, pp. 162–172.

[24] 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)*. ACM Press, July 2002, pp. 467–474.

[25] A. Oram, Ed., *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. Sebastopol, CA: O'Reilly & Associates, 2001.

Pınar Yolum is an assistant professor at Boğaziçi University, Istanbul. She received her PhD and MS in computer science from North Carolina State University in 2003 and 2000, respectively, and her BSc in computer engineering from Marmara University, Istanbul in 1998. She worked as a post-doctoral researcher in the Free University of Amsterdam. Her research interests include multiagent systems, peer-to-peer networks, and service-oriented computing.

Munindar P. Singh is a professor at North Carolina State University. His research interests include multiagent systems and Web services, where he specifically addresses the challenges of trust, service discovery, and business processes and protocols in large-scale open environments. Munindar's recent books include the coauthored *Service-Oriented Computing* and the edited *Practical Handbook of Internet Computing*.

Munindar was the editor-in-chief of *IEEE Internet Computing* from 1999 to 2002 and continues to serve on its editorial board. He is also a member of the editorial boards of the *Journal of Autonomous Agents and Multiagent Systems* and the *Journal of Web Semantics*, and serves on the steering committee for the *IEEE Transactions on Mobile Computing*.