# Emergent Properties of Referral Systems

Pınar Yolum
pyolum@eos.ncsu.edu

Munindar P. Singh
singh@ncsu.edu

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA

## ABSTRACT

Agents must decide with whom to interact, which is nontrivial when no central directories are available. A classical decentralized approach is referral systems, where agents adaptively give referrals to one another. We study the emergent properties of referral systems, especially those dealing with their quality, efficiency, and structure. Our key findings are (1) pathological graph structures can emerge due to some neighbor selection policies and (2) if these are avoided, quality and efficiency depend on referral policies. Further, authorities emerge automatically and the extent of their relative authoritativeness depends on the policies.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

## General Terms

Experimentation, Performance, Security

## Keywords

Emergent properties; referrals; PageRank

## 1. INTRODUCTION

Consider multiagent systems consisting of autonomous agents. Imagine that our agents represent *principals* who could be people or businesses providing and consuming *services*. The services could involve serving static pages, processing queries, or carrying out e-commerce transactions, but their details are not represented in this paper. The following aspects of real systems are relevant here:

- The agents offer varying levels of trustworthiness and are interested in finding other trustworthy agents. They track each other's trustworthiness. The agents can judge the quality of a service obtained and adaptively select their neighbors in order to improve their local performance. When requested, an agent may provide a service or give a referral. Importantly, by giving and taking *referrals*, agents can cooperate with one another to find trustworthy agents with whom to interact. Thus, they form a *referral system*. Notice that trust applies both to the ultimate service provider and to the agents who contribute referrals to that provider.
- The agents are autonomous. That is, an agent may or may not respond to another agent by providing a service or a referral. When an agent does respond, there are no guarantees about the quality of the service or the suitability of a referral. Likewise, we do not assume that any agent should necessarily be trusted by others: an agent unilaterally decides how to rate another principal.

The above model addresses the important challenge of finding trustworthy agents, which is nontrivial in open systems. First, referrals can apply even in the absence of centralized authorities and even when regulations may not ensure that services are of a suitable quality. Second, because service needs are often context-sensitive, a response from an agent can potentially benefit from the knowledge that it has of the other's needs.

**Contributions.** The idea of referrals in multiagent systems goes back a long time. Referrals have been used in specific applications (see Section 5). However, we treat referrals as the key organizing principle for large-scale multiagent systems. Our objective in this paper is to study the conceptual aspects of referral systems, especially with respect to their emergent properties. Consequently, we first study how referral policies influence the quality and efficiency of referral systems and how these policies influence the way authorities emerge.

Next, we study the structural properties of referral systems. Recent work has studied the structure of the Web as it happens to have emerged mostly through links on human-generated, static pages. Whereas existing work takes an after-the-fact look at Web structure, we can study the emerging structure of a referral system as it relates to the local policies of its members. Links over which parties request or give referrals and the referrals they give induce a natural structure on a referral system, leading to two important consequences. First, major application classes can be modeled via different structures. Second, the structure evolves in interesting ways based on the policies followed by the different parties. To this end, we identify pathological graph structures that can emerge.

**Organization.** Section 2 gives additional details on our model of referrals among autonomous agents, possible applications domains, and our experimental setup. Section 3 introduces metrics to measure quality of networks, and evaluates the performance of agent policies. Section 4 characterizes some possible structures of the network in terms of their emergence and desirability for referral systems. Section 5 discusses the relevant literature and motivates directions for further work.

## 2. TECHNICAL FRAMEWORK

The agents act in accordance with the following abstract protocol. An agent begins to look for a trustworthy provider for a specified service. The agent queries some other agents from among its *neighbors*. A queried agent may offer to provide the specified service or, based on its *referral policy*, may give referrals to other agents. The querying agent may accept a service offer, if any, and may pursue referrals, if any. Each agent maintains models of its acquaintances, which describe their *expertise* (i.e., quality of the services they provide) and *sociability* (i.e., quality of the referrals they provide). Both of these elements are learned based on service ratings from its principal. Using these models, an agent applies its *neighbor selection policy* to decide on which of its acquaintances to keep as neighbors. Key factors include the quality of the service received from a given provider, and the resulting 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. An agent's own requests go to some of its neighbors. Likewise, an agent's referrals in response to requests by others are also given to some of its neighbors, if any match. This, in a nutshell, is our basic social mechanism.

Together, the neighborhood relations 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.

### 2.1 Applicable Domains

This framework enables us to represent different application domains naturally. Two important domains are commerce and knowledge management, which have differ in their notions of service and how the participants interact.

In a typical commerce setting, the service providers differ from the service consumers. The service consumers lack the expertise in the services that they consume and usually, no matter how much they use the services, their expertise doesn't 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. However, you can still evaluate 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 knowledge of the domain.

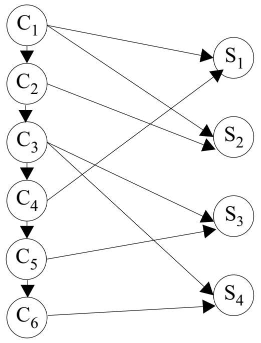

**Figure 1: A schematic configuration for e-commerce**

By contrast, in knowledge management, the idea of "consuming" knowledge services would correspond to acquiring expertise in a given domain. 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. Thus, they could increase their expertise over time, and possibly answer queries from others [20]. Following the same intuition, the questions an agent generates would also depend on its expertise to ensure that the agent doesn't ask a question whose answer it already knows.

Figure 1 is 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. Consumers are connected to each other as well as to the service providers. Consumers have high *interest* in getting different types of services, but they have low expertise, since they don't offer services themselves. These links are essentially paths that lead to service providers with different expertise. In this model, the service providers are dead ends: they don't have outgoing edges, because they don't initiate queries or give referrals. In other words, providers have low sociability. Their true and modeled expertise may of course be high. The interests and expertise of the agents are represented as term vectors from the vector space model (VSM) [12], each term corresponding to a different domain.

### 2.2 Evaluation Metrics

The relevant global properties of the system that we study here are formally characterized by some metrics, usually involving vector operations.

**Similarity.** To capture the *similarity* between an agent and a query, we seek a formula that is commutative, i.e., a vector $I$ is as similar to $J$ as $J$ is to $I$. A common similarity measure is the cosine of the angle between two vectors, but the cosine of the angle between two vectors does not capture the effect of their length. Since the two vectors will always be in the first quadrant, our formula does not consider the angle between the two vectors explicitly. The following formula captures the Euclidean distance between two vectors and normalizes it to get a result between 0 and 1. It also applies in measuring the similarity of the members in a group based on their interests. ($I$ and $J$ are of length $n$.)

$$I \oplus J = \frac{e^{-\|I-J\|^2} - e^{-n}}{1 - e^{-n}} \quad (1)$$

**Capability.** The *capability* of an agent for a query measures how similar and how strong the expertise of the agent is for the query [14]. Capability resembles cosine similarity but also takes into account the magnitude of the expertise vector. That is, agents that have expertise vectors with greater magnitude are more capable for the query vector. In Equation 2, $Q$ ($\langle q_1 \ldots q_n \rangle$) is a query vector, $E$ ($\langle e_1 \ldots e_n \rangle$) is an expertise vector and $n$ is the number of dimensions these vectors have.

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

**PageRank.** PageRank is a metric used by Google to rank Web pages that are returned for a query [4]. The PageRank of a Web page measures its authoritativeness. Informally, 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. We use the same metric to measure the authoritativeness of agents. The PageRank of an agent is calculated using Equation 3, where $P(i)$ denotes the PageRank of agent $i$, $I_i$ denotes agents that have $i$ as a neighbor, and $N_j$ denotes the agents that are neighbors of $j$. The PageRanks are normalized using a constant $d$, where $d$ is taken to be 0.85 as in the original paper [4].

$$P(i) = d \sum_{j \in I_i} \frac{P(j)}{N_j} + (1 - d) \tag{3}$$

## 2.3 Referral Algorithms

We have implemented a distributed platform using which adaptive referral systems for different applications can be built. However, we investigate the properties of interest over a simulation, which gives 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, where between 10% to 25% of the agents are service providers in one domain, and the remaining agents are consumers. The interests of the consumers can span several domains. Each agent has 8 neighbors and is initialized with the same model for each neighbor. This initial model encourages the agents to query their neighbors.

**Algorithm 1** Ask-Query()

```
 1: Generate query
 2: Send query to matching neighbors
 3: while (!timeout) do
 4:     Receive message
 5:     if (message.type == referral) then
 6:         Send query to referred agent
 7:         Add referral to referral graph
 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
```

An agent that is generating a query follows Algorithm 1. 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. 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). Next, the agent sends the query to a subset of its neighbors (line 2). The main factor here is to determine which of its neighbors would be likely to answer the query. We usually determine this through the capability metric.

An agent that receives a query acts in accordance with Algorithm 2. An agent answers a question if its expertise matches a question. If the expertise matches the question, 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.

**Algorithm 2** Answer-Query()

```
1: if hasEnoughExpertise then
2:     Generate answer
3: else
4:     Refer neighbors
5: end if
```

Back in Algorithm 1, if an agent receives a referral to another agent, it sends its query to the referred agent (line 6) and adds a referral link to its *referral graph* (line 7). Simply put, a referral graph is a directed graph where the nodes denote agents and an edge denotes that the source of the edge has referred to the target of the edge. Each agent builds a referral graph for each query it has generated. After an agent receives an answer, it evaluates the answer by computing how much the answer matches the query (line 13). Thus, implicitly, the agents with high expertise end up giving the correct answers. After the answers are evaluated, the agent uses its *learning policy* to update the models of its neighbors (line 14). 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. 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 might have to drop some neighbors as well.

Service consumers search for trustworthy service providers. Since the whole society can be viewed as a graph, the search for a service provider is essentially a search starting from a consumer node, which may terminate at a provider node. In this respect, the search might look trivial and could be performed with any standard search algorithm. However, there are two major challenges. First, each agent in the system has a partial view of the graph. For example, in Figure 1, $C_2$ knows that $C_3$ and $S_2$ are its neighbors, but may not know that $S_4$ is $C_3$'s neighbor. Second, each agent in the graph is autonomous and may well have unique policies to take care of different operations like 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, $C_2$ may ask $C_3$ but if $C_3$ does not respond, then the search path becomes a dead-end.

With only incomplete information and possible non-responsive agents, what is a good strategy to follow in order to find the desired service providers? Obviously, the answer depends on many variables, including the size of the population, the interest and expertise of the agents, the policies agents follow, and so on. Here, instead of studying the individual strategies of the agents, we study the global properties of the system that emerge as a result of some agent policies. The properties we emphasize are the effectiveness and the structure of the networks. For both cases, we evaluate the influence of these properties on service location with some simplistic policies.

## 3. EFFECTIVENESS AND EFFICIENCY

The effectiveness of a system measures how easily agents find useful providers. We study two metrics to measure the effectiveness of the system: direct quality and $n$th best quality. Both metrics are defined as obtained by an agent and then averaged over all agents.

The *direct quality* viewed by an agent reflects, via (2), the usefulness of the neighbors of the agent, given its interest and their expertise. That is, we estimate the likelihood of the neighbors themselves giving good answers to the questions and ignoring the other agents.

Next, we take into account an agent's neighbors and other agents. 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 from the agent, the less their contribution to the quality seen by the agent. The contribution of agent $j$ to agent $i$'s quality is given by:

$$\frac{I_i \otimes E_j}{path(i, j)} \tag{4}$$

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 *nth best* measure. That is, we measure the quality obtained by an agent by its $n$th best connection in the network. The choice for $n$ is tricky. If $n$ is too big, each agent's quality is equally bad. On the other hand, 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 $n$th best metric to measure an agent's quality and take $n$ to be twice the number of neighbors the agent has.

## 3.1 Effectiveness

A referral policy specifies to whom to refer. We consider some important referral policies. We tune the simulation 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 very low. This resembles Gnutella's search process where each servent forwards an incoming query to all of its neighbors if it doesn't already have the requested file [6].

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 [8].

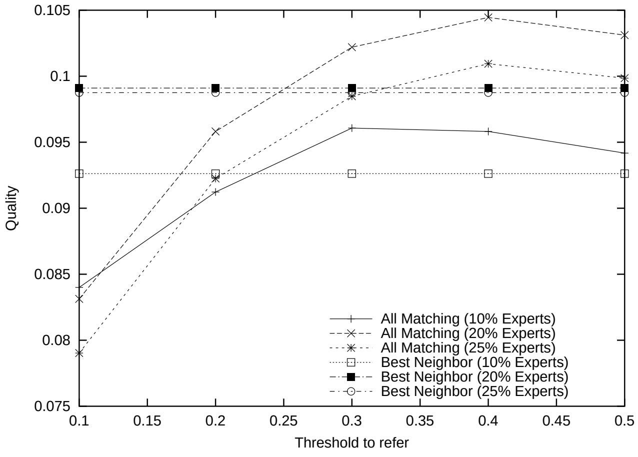

**Figure 2: Performance of referral policies**

We test the performance of different policies by varying the capability threshold. Figure 2 plots this threshold versus the quality of the graph for different policies. We plot different populations on this graph varying the percentage of experts in the population. There are three populations, each with 400 agents but with 10%, 20%, and 25% experts in them. Each agent generates eight queries during a simulation run, resulting in 3200 queries all together. Each agent is neighbors with two percent of the population, which in this case is eight agents. Each agent sends its query to its neighbors. The neighbors then apply the selected referral policy. Thus, based on the referral policy, each query results in different number of agents being contacted. We limit the length of the referral graphs to five—similar to Gnutella's time-to-live value.

In Figure 2, the lines marked *All Matching* show *Refer all matching* policy for varying thresholds on the $x$ axis. The case where the referral threshold is set to $0.1$ denotes the *Refer all* policy. The lines marked *BestNeighbor* plot the *Best Neighbor* policy, which is independent of the threshold.

OBSERVATION 1. *Among these referral policies Refer all matching results in graphs with higher quality, where the best threshold increases with the percentage of experts in the society.*

Among the three policies, *Refer all* performs the worst for all three populations. As seen in Figure 2, when agents use this policy, the quality never becomes more than $0.085$. The *Best Neighbor* policy performs better than *Refer all matching* policy for small values of the capability threshold (e.g., $0.2$). For thresholds greater than $0.2$, the *Refer all matching* policy performs better than the *Best Neighbor* policy.

## 3.2 Efficiency

This quality metrics introduced above are optimistic, since 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 by. To illustrate this point, we measure the efficiency of finding answers. Efficiency is defined as the ratio of the correct answers received to the number of agents contacted. Figure 3 plots the capability threshold versus the efficiency for different referral policies.

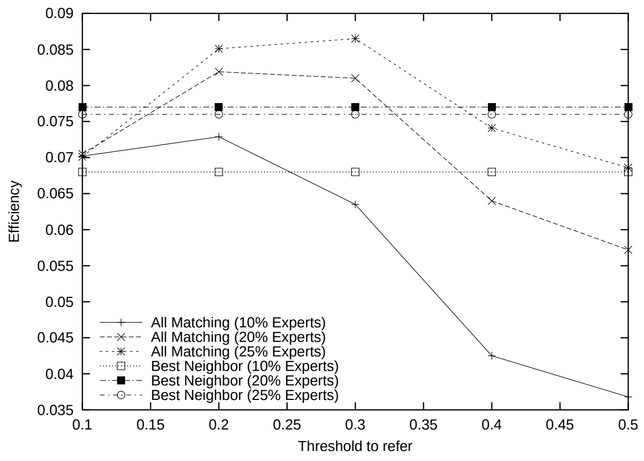

**Figure 3: Effect of responsiveness on performance**

OBSERVATION 2. *Among these referral policies Refer all matching finds providers with the highest ratio, where the best threshold increases with the percentage of experts in the society.*

When the threshold is set low, the referrals becomes less selective, and thus many agents are contacted. Conversely, when the threshold is set high, the referrals are too selective, and not enough agents are contacted to find useful answers. Hence, both extremes of the threshold suffer from lower efficiency.

Even though the *Refer all matching* performs well for both effectiveness and efficiency, different matching thresholds perform better in each case. For example, a threshold of $0.4$ results in the best quality graph for the population with $20\%$ experts, whereas the actual ratio of good answers received with this threshold is lower than most of the thresholds. In other words, getting close to a useful agent does not guarantee receiving useful answers.

## 3.3 Authoritativeness

Some agents are identified as authoritative as a result of their being chosen as neighbors by other authoritative agents. Presumably, authoritative agents are the most desirable to interact with. In general, agents with high expertise or high sociability would be candidates for authorities. We measure the authoritativeness of each agent using the PageRank metric (Equation 3) and study the effect of referral policies in the emergence of authorities.

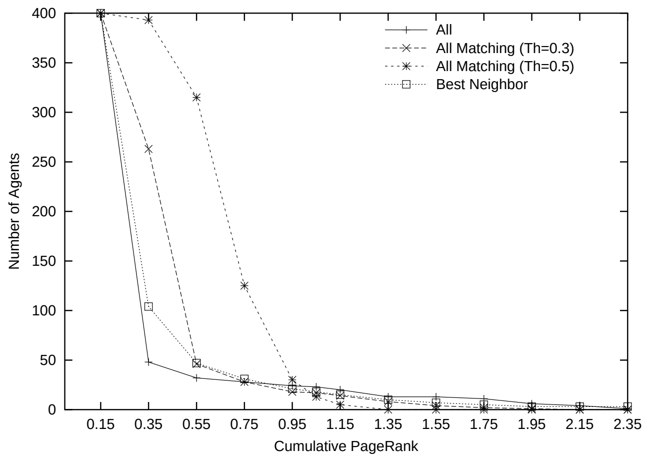

**Figure 4: PageRank distribution**

Figure 4 plots the number of agents that achieve greater than a given PageRank for different referral policies. We only show the plots for populations with $10\%$ providers, although the curves for other populations are similar. If the agents use the *Refer all* policy, few authorities with high PageRanks emerge. For example, only $48$ agents have a PageRank greater than $0.35$. Compare this to the case where agents use the *Refer all matching* policy where the referral threshold is $0.5$. In that case, $393$ agents get a PageRank higher than $0.35$, and even $125$ agent get a PageRank higher than $0.75$. On the other hand, with this threshold the highest PageRank achieved is $1.15$, wheres the highest PageRank achieved with the *Refer all* policy is $2.35$.

OBSERVATION 3. *While more authorities emerge through Refer all matching policies, Refer all policy causes emergence of authorities whose level of authoritativeness is higher.*

## 4. NETWORK STRUCTURE

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.

**Neighbor selection policies.** 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? At one extreme, if each agent chooses the best providers it knows as neighbors, then the graph would acquire several stars each centered on an agent who is the best provider for the agents whose neighbor it is. On the other hand, if everybody becomes neighbors with agents that have slightly more expertise than themselves the structure will tend to be a tree, similar to an organizational hierarchy. 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.

- *Providers.* Sort acquaintances by how their expertise matches the agent's interests.
- *Sociables.* Sort acquaintances in terms of sociability.
- *Weighted average.* Sort acquaintances in terms of a weighted average of sociability and how their expertise matches the agent's interests.

### 4.1 Bipartite Graphs

Consider a bipartite graph. 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 on the sole basis of their local knowledge.

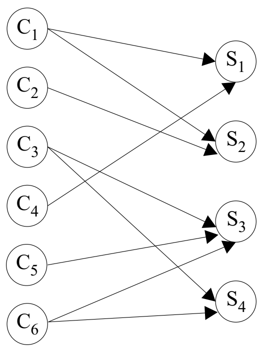

**Figure 5: Consumers don't help each other**

OBSERVATION 4. *The quality of a bipartite graph is stable and suboptimal.*

Since the service providers do not have outgoing edges, they will not refer 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 other agents that it cannot reach. Configurations that allow reachability to these agents will have better quality. Thus, the quality of the bipartite graph is not optimal.

Even if the graph is not bipartite, the structure could be very close to a bipartite graph. Let's say that the graph would be bipartite if we took out a few edges from the graph. This is still dangerous, since the graph might quickly evolve into a bipartite graph. The number of edges needed to be removed is a metric for determining the structural quality of the graph.

Obviously, we need to prevent the graph from turning into a bipartite graph. The only way to do so is if the agents choose their neighbors in a certain manner so as to ensure that these structures are not realized. Accordingly, we study the neighbor selection policies to see if they can cause the graph to turn into a bipartite graph.

OBSERVATION 5. *In a population where 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.*

When the agents use *Providers* policy, convergence to a bipartite graph is unavoidable, because as each agent discovers the service providers in the society, it will replace its sociable neighbors with the providers that it finds. While this is the case for the *Providers* policy, the same effect does not hold for *Weighted Average* or *Sociables* policies, since with these policies consumers may choose some other consumers as neighbors, because of the consumers' sociability.

## 4.2 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 [19]. Thus, different components have disjoint vertices and are mutually disconnected. Consequently, consumers can at best find service providers in their own components. In other words, 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. Since the consumers are the only sociable agents, consumers link up with other consumers only. In the worst case, this results in the providers being totally isolated from the consumers.

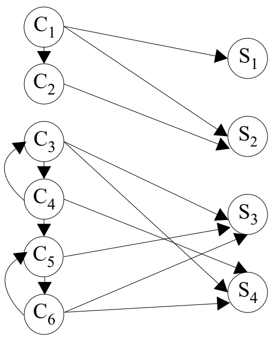

**Figure 6: Some consumers cannot reach some providers**

OBSERVATION 6. *In a population where each agent exercises the Sociables policy, the graph ends up with more than one weakly-connected component.*

## 4.3 Clustering

We define a clustering coefficient to measure if similar agents become neighbors. Our coefficient is similar in motivation to Watts' cliquishness coefficient [18]. However, we also take into account how similar the agent itself is to its neighbors. The average of all the agents' clustering coefficients constitutes the clustering coefficient of the graph. The reflexive interest clustering $\gamma(i)$ measures how similar the interest vectors of an agent $i$'s neighbors (including $i$ itself) are to each other. Below, $N_i$ denotes the set consisting of node $i$ and all its neighbors. $E_i$ denotes all the edges between the nodes in $N_i$.

$$\gamma(i) = \frac{\sum_{(i,j)\in E_i} I_i \oplus I_j}{|N_i|(|N_i|-1)} \quad (5)$$

Figure 7 plots the quality of the network for different values of reflexive interest clustering. Each plot corresponds to a different neighbor selection policy. $W$ denotes the weight of the sociability in choosing a neighbor. When $W$ is set to 0, the Providers policy is in effect. When $W$ is set to 1, the Sociables policy is in effect. Other values of $W$ measure weighted averages of the sociability and expertise. In our simulation, each agent selects neighbors after every two queries, and each simulation is run for four neighbors changes. The four points on the plot lines correspond to each neighbor change.

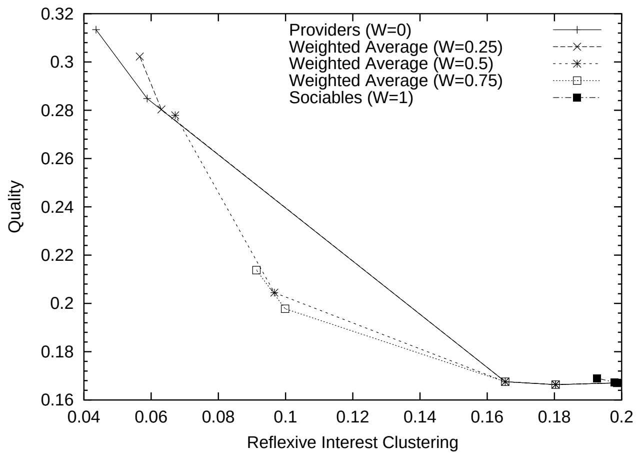

**Figure 7: Quality versus reflexive interest clustering**

OBSERVATION 7. *Reflexive interest clustering decreases with an increase in quality.*

An increase in quality shows that some consumers are getting closer to the qualified service providers. This decreases the reflexive interest clustering since now all those clustered consumers can get to the service provider through referrals and no longer need to be neighbors with other similar consumers. Consider a group of travelers who are not aware of a qualifier travel agent. As soon as one of them discovers it, the quality of the network will increase. Further, it will refer this new travel agent to its neighbors when asked for, affecting the neighbors to eventually point to the travel agent. This will decrease the interest clustering of that particular group of travelers.

Reflexive interest clustering has an interesting consequence: congestion. An agent in the network is *congested* when the number of incoming edges is large. The idea of congestion here resembles the one in computer networks. In computer networks, if there are more packets coming into a node than the ones leaving the node, then the node will be congested [16]. However, here we are not concerned about the out-degree, since the out-degree is not representative of how much of the incoming traffic is handled properly.

In Figure 8, the $x$ axis shows the in-degree and the $y$ axis shows the number of agents. Each box corresponds to the number of agents that have an in-degree greater than the given in-degree value.

The dashed lined box shows the initial distribution. That is, initially approximately $50$ agents have an in-degree greater than $10$, but none of the agents have an in-degree more than $20$. Contrast this to the distribution after interest clustering takes place, shown with solid boxes. There are five agents whose in-degree is greater than $80$ and two of these even have in-degree greater than $90$.

OBSERVATION 8. *Increasing reflexive interest clustering increases congestion.*

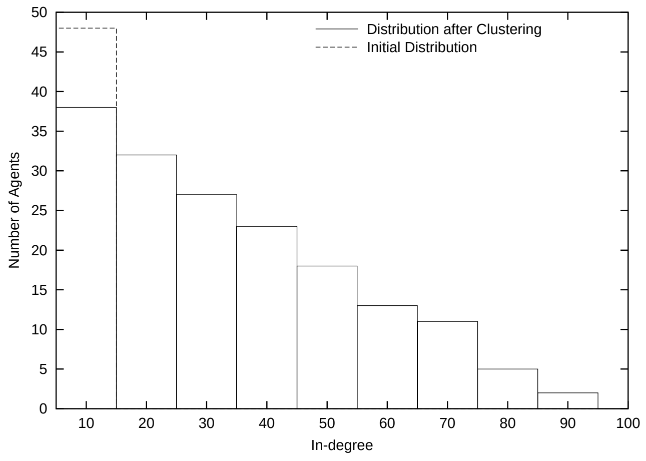

**Figure 8: Distribution of in-degree before and after clustering**

A cluster of agents with similar interests will want to be neighbors with the same service provider. This is analogous to the case where a group of agents who are interested in traveling find an expert in travel agencies. All the enthusiastic travelers want to ask their questions to the same expert, making the travel expert congested.

Following congestion is an immediate increase in direct quality metrics. If an agent is congested, then a number of consumers are using this agent heavily. This results in a high direct quality; most consumers are happy with their immediate neighbors. But interestingly, the global quality might not be equally well, as can be seen in Figure 9, which plots direct quality metric versus the quality metric for different neighbor selection policies.

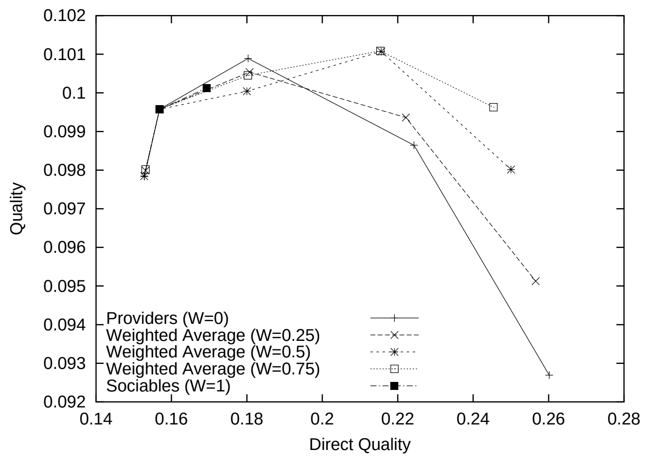

**Figure 9: Correlation between direct quality and quality**

OBSERVATION 9. *High average direct quality does not guarantee a high global quality.*

Each agent tries to maximize its own welfare by its choice of neighbors. A particular set of neighbors might provide better answers than others, but this does not ensure that all other agents are better off. Consider an extreme case, where each agent has only one neighbor. Agent $A$ and agent $B$ establish a reciprocal relation. Although both of them could be a good match for each other, they become isolated from the rest of the agents. Although, locally they might have made a correct decision, globally the quality of the whole graph will go down.

## 5. DISCUSSION

Our approach enables us to study the emergent structure of multiagent systems as they are employed to help participants jointly discover and evaluate services. Below, we discuss some related approaches and then consider the greater goals of our work and the directions in which it can expand.

Referrals capture the manner in which people normally help each other find trustworthy authorities [9]. This is an important motivation for referral systems. Multiple Intelligent Node Document Servers (MINDS) was the earliest agent-based referral system [3]. 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 [7]. Our work, on the other hand, 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 [21]. 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.

Gibson *et al.* discuss an approach to infer web communities from the topology of links among web pages [5]. Communities here are defined in terms of related sets of *hubs*, which ideally point at lots of authorities, and *authorities*, which are ideally pointed by lots of hubs. The main difference between previous work and our approach is that our model is inherently heterogeneous, whereas previous work treats all pages as essentially alike. Also, web-pages are vivid in that what you see is what you get, whereas services in general leave a lot of room for confusion and misunderstanding, thus increasing the importance of trust. In this sense, our work generalizes over the previous research. It would be interesting to see how the algorithms, such as of Gibson *et al.*, can be extended to apply in our model.

Wang develops an approach for organizing agents into communities based on the similarity of their interests and expertise [17]. 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 7, i.e., clustering does not favor quality, will dominate.

Adamic *et al.* study different local search algorithms in power-law networks to exploit the advantages of having nodes with high out-degrees [2]. Their local search strategies are analogous to our referral policies. One of their strategies is to send the message to the neighbor with the most outgoing edges, assuming each node is aware of the number of outgoing edges of their neighbors. This resembles our concept of sociability. Adamic *et al.*'s approach chooses a peer with high out-degree because it will allow the message to get to more peers. In our case, we are not concerned about maximizing the number of agents but (on the contrary) optimizing that each message reaches agents with sufficient expertise. Thus, sending messages to highly sociable agents ensures that these sociable agents will find the agents with sufficient expertise.

Shehory develops a decentralized approach for locating agents where agents find one another with the help of neighbors [13]. Rather than returning referrals as here, the neighbors themselves look for the desired agent. This is similar to Gnutella's search mechanism. Shehory shows how increasing the average path length can increase the efficiency of agent location. Our approach increases the efficiency of the search through agent policies. 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 aspects are not directly addressed in Shehory's approach.

Recently, several peer-to-peer network architectures have been proposed [15, 11, 1]. Essentially, these systems model the network as a distributed hash table where a deterministic protocol maps keys to peers. Thus, given the key of an item, there is a unique peer that is responsible for holding that item. The peers are not autonomous—they don't choose their data items, but the data items are assigned to them. Each peer has a table that aids the search when the item being searched does not reside at this peer. This is similar to our neighbors concept. However, in our approach, the neighbors are chosen based on how well they match a given agent; each agent can change its neighbors at will. Conventional systems lack this kind of adaptability. First, the peers in the tables are defined deterministically. Second, peers cannot change their neighbors, unless the neighbors get off-line.

In our future work, we plan to explore the relationships between various policies and performance further, especially in the context of the structural assumptions of different applications.

## 6. ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under grant ITR-0081742. This is a revised and much extended version of a paper appearing in the AAMAS 2002 Workshop on Agents and Peer-to-Peer Computing. We thank the anonymous reviewers for helpful comments.

## 7. REFERENCES

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *Proceedings of Cooperative Information Systems (CoopIS)*, pages 179–194, 2001.

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physics Review E*, 64(46135), 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] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring Web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space - Structure in Hypermedia Systems*, pages 225–234. ACM, 1999.

[6] G. Kan. Gnutella. In *[10]*, chapter 8, pages 94–122. 2001.

[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] A. Langley. Freenet. In *[10]*, chapter 9, pages 123–132. 2001.

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

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 161–172. ACM, 2001.

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

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

[14] M. P. Singh, B. Yu, and M. Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, Apr. 2001.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 149–160. ACM, 2001.

[16] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1996.

[17] F. Wang. Self-organising communities formed by middle agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1333–1339. ACM Press, July 2002.

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

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

[20] P. Yolum and M. P. Singh. Flexible caching in peer-to-peer information systems. In *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS)*, pages 72–83, 2002.

[21] B. Yu and M. P. Singh. Searching social networks. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, July 2003. To appear.