Abstract

YOLUM, PINAR. Properties of Referral Networks: Emergence of Authority and Trust (Under the direction of Munindar P. Singh).

Developing, maintaining, and disseminating trust in open environments is crucial. We develop a decentralized approach to trust in the context of service location. Service providers and consumers are modeled as autonomous agents participating in a multiagent system that functions as a referral network. When a service is requested, an agent may provide the requested service or give a referral to another agent. The agents can judge the quality of service obtained. Importantly the agents can adaptively select their neighbors, decide with whom to interact, and choose how to give referrals.

The agents' actions lead to the evolution of the referral network. We study the emergent properties of referral networks, especially those dealing with their quality, efficiency, and structure. We first show how the exchange of referrals affect locating service providers, then identify undesirable network structures and show under which conditions these network structures emerge.

When agents refer and change neighbors in certain specific ways, based on link structure, some agents are identified to be substantially more popular or authoritative than others. These asymmetric distributions of popularity and authoritativeness resemble those seen on the Web. A referral corresponds to a customized link generated on demand by one agent for another. Referrals thus yield a basis for studying the processes underlying authority, especially as they affect the structure of the evolving social network of agents. Whereas existing work takes an after-the-fact look at Web structure, we can study the emergence of structure as it relates to the policies of the members.

Further, we propose a graph-based representation of services that can be applied in conjunction with our referrals-based approach. This representation captures natural relationships among service domains and provides a simple means to accommodate the accrual of trust placed in a given party. We study this representation in depth, especially in terms of factors that apply to trust.

The properties uncovered through this dissertation can serve as guidelines to develop robust referral systems that are both efficient and effective.

PROPERTIES OF REFERRAL NETWORKS: EMERGENCE OF AUTHORITY AND TRUST

BY

PINAR YOLUM

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

RALEIGH

JUNE 2003

APPROVED BY:

DENNIS R. BAHLER

CARLA D. SAVAGE

MUNINDAR P. SINGH

(CHAIR OF ADVISORY COMMITTEE)

PETER R. WURMAN

Trust I seek, and I find in you
Every day for us something new
Open mind for a different view
And nothing else matters

—Metallica

To İlker

Biography

Pınar Yolum was born on April 23rd, 1976 in Istanbul, Turkey. She got her Bachelor of Engineering degree in Computer Engineering from Marmara University in 1998 and her Master of Science degree in Computer Science from North Carolina State University in 2000. She has been a Ph.D. student in the Computer Science Department of North Carolina State University since then.

Acknowledgements

Many people have contributed to the making of this dissertation.

I was privileged to have Dennis Bahler, Carla Savage, and Peter Wurman on my Ph.D. committee. They gave numerous useful suggestions that helped to improve this dissertation. My adviser, Munindar Singh, worked with me closely, understood my thoughts before I understood them myself, and always encouraged me to work more. Mike Eberdt patiently proofread many of my writings over the years, including this dissertation. The National Science Foundation supported this work under grant ITR-0081742.

Mike Eberdt, Emmanuel Sanchez, Selene Leon, Mine Haşhaş, Simla Tokgöz, Demet Başal, Bin Yu, Zhengang Cheng, Jie Xing, Ashok Mallya, Raghu Sreenath, Michael Maximilien, Srivatsa Chivukula, Subhayu Chatterjee, Soydan Bilgin, Ergun Biçici, Amit Chopra, and Yathi Udupi generously offered their friendship.

İnci Demirkanlı, Gökçe Gemici, İlker Birbil, Oya Yolum, Başak Yolum, Ceren Belge, İmge Ceranoğlu, Gökhan Çiftçi, Serhat Özcanlı, Özge Kocabayoğlu, Mehmet Can Yıldız, Onur Demir, Gürhan Küçük, Cenk Erdil, Nurhan Çetin, and Mike Eberdt came to visit at times when I felt I could not live another day.

My family called frequently and made sure I was all right. They always reminded me of where I belong and what is important in life. I shared everything Raleigh could offer with İlker. Without him, I would not have been brave enough to pursue this degree.

Table of Contents

List of Figures

List of Symbols

i, j, k, l, u, v : Agents

P_i : Provider i

C_i : Consumer i

C, D : Communities

C_i : Rank of i in community C

$P(i)$: PageRank of i

K_i : Set of agents that have i as neighbor

N_i : Set of agents that are neighbors of i

V_i : Set of agents containing i and all of i's neighbors

M_i : Set of edges between the agents in V_i

d : PageRank coefficient

$\sigma_{u,v}$: v's sociability from u's point of view

I_i : i's interest vector

E_i : i's expertise vector

$S(i)$: Sociability rank of i

$\rho(C, D)$: Spearman's correlation between C and D

η : Number of edges between consumers

$\gamma(i)$: Interest similarity of i

W : Weight of sociability in choosing a neighbor

R : Linear regression correlation coefficient

$P(i)$: In-degree of i

$R(i)$: In-degree rank of i

α : Power law exponent

s_i : Service type i

S_i : Provider i

t_{pi} : Trustworthiness of p at i

w_{ij} : Weight of edge ij

θ : Promotion threshold

λ : Required number of interactions before promotion

Chapter 1

Introduction

The Web is moving from a collection of pages to a collection of distinct entities that provide and use *services*. The entities can be people or businesses, each potentially supported by an automated assistant. Each service can involve tasks that vary from serving information such as Web pages to performing other complex tasks that involve carrying out transactions or processing queries or even composing other services. The services are not merely distinguished by the domain or the tasks associated with them, but also in terms of any other features of interest, such as the price or the speed of a task involved in the service. Hence, a service is described as aggregating multiple features.

The entities that *provide* the services may differ in the resources they own, the service types they provide, the way they carry out their services, and and the qualities of the services for various features. Likewise, the entities who use or *consume* the services vary in their needs and evaluations of services. This flexibility of operations and interactions can only be realized if the entities are autonomous and heterogeneous. Hence, it is crucial to allow entities to retain their heterogeneity and enable them to operate autonomously [Singh, 2003].

The entities can exercise their autonomy to decide what actions they want to perform, with whom they want to interact, or how they want to carry out their tasks. From a service consumer's point of view, this implies that it 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. Similarly, just because a consumer wants to carry out interactions with a provider does not mean that the provider has to agree. The service providers can autonomously decide whom they want to serve or the quality of service they want to provide. Providers can serve the different needs of the different consumers with a varying quality.

The above model is significantly and substantially different from traditional object-oriented architectures where objects invoke methods on others and each object is expected to respond in the previously agreed manner. The autonomy of the entities precludes interactions through method invocation as in traditional programmatic interfaces. Instead, entities exchange messages to request, negotiate, and fulfill information needs.

The entities can be heterogeneous. That is, entities can be diverse in their specifications and adopt different strategies to carry out their interactions. This implies that, in general, specifics of service implementations are not revealed to the consumers, nor to any other external parties. Not knowing the service implementation makes it harder to judge the quality of a service for two reasons. One, mechanisms based on a third-party evaluation of a service implementation cannot be employed. Two, because each consumer is different, a third party cannot evaluate service outcomes either. Consequently, each consumer must evaluate the service it receives.

Because the services are autonomous and heterogeneous, finding the right service providers is a significantly greater challenge in large scale open systems than in traditional distributed systems. The scale and dynamism of such large systems imply that a participant would not know and would not be able to keep up with all potentially relevant participants. Truly open and large systems depart 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*.

1.1 Trust

To study trust more rigorously, it is necessary to understand various properties of trust. Trust is established between a *trustor* and a *trustee* with respect to a context. For purposes of the present research, the necessary aspects of context are reflected in the services being sought and provided.

Trust is inherently for a purpose and spans multiple dimensions. A trustee may be competent in some services but not in others. Accordingly, a trustor would (or would not) trust a trustee 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 party, one must clearly formulate the service or individual actions in question. Even when these are made explicit, two trustors who interact with the same trustee may have different assessments of the trustee's trustworthiness. This variance could occur because of different evidence or different evaluations of the same evidence.

Security is an essential first step in establishing trust. Classical security techniques address the problems of authentication, authorization, and nonrepudiation. These techniques help ensure that both the trustor and trustee are aware of whom they are dealing with. However, this certainly is not enough to establish trust between the entities. Knowing another party's identity and checking its authorizations does not guarantee that it will act in a manner that is in your best interest or even in a manner that does not exploit your interests unfairly.

More specifically, even after a consumer locates a provider and they agree to interact, the *risk* remains that the provider may fail to meet the expectations of the consumer in terms of service quality. Some possible reasons for this failure are that the service provider may not be competent to perform the task, or it may be cheating on purpose. Independently of the reason, once the consumer commits to interact with the provider, there is always the risk of receiving an unacceptable quality of service. Interestingly, the risks that arise in this context all result from the consumer's *commitment* to interact with the provider. The existence of a commitment with a potential risk of being violated or even poorly fulfilled leads to the need for trust. Often, trust is seen as a remedy to decrease this risk [Luhmann, 2000].

Trust can be established through three major means: institutional, local, and social. *Institutional* trust, or trust in authoritative institutions or organizations, is common in the off-line world. People trust in the power of these institutions to stabilize their interactions [Misztal, 1996, p. 26]. For example, if a business is registered with a government agency, you might trust the agency to monitor the business as well as enforce restrictions or procedures that protect your benefits. These agencies make the trustee accountable for any misdeeds [Dasgupta, 2000]. Also, institutional trust in an organization may arise because of the organization's prestige. Even if such an organization does not enforce any rules on its members, a business could be trusted by others for belonging to the organization.

Current distributed trust management approaches can be thought of as formalizing institutional trust, because they assume that digital certificates issued by various certificate authorities lead to trust [Castelfranchi and Tan, 2001]. That is, these approaches usually assume that trust is established merely through a chain of endorsements beginning with some trusted authority [Grandison and Sloman, 2000; Kagal et al., 2001; Wong and Sycara, 1999]. However, only the most trivial level of trust (e.g., endorsements) can be established through such a mechanism. For example, knowing that a Web site carries a digital certificate issued by another known site does not guarantee that the Web site will act in a trustworthy manner.

For this reason, artificial intelligence approaches take an empirical stance on trust, attempting to create trust based on evidence of some sort. The evidence could be *local* or *social*. Local trust means considering previous direct interactions with a trustee (i.e., local evidence), which often are the most valuable in creating trust for the following reasons. One, since the trustor itself evaluates the interactions, the results are more reliable. Two, the context in which the trustworthiness of the provider is evaluated is explicit and relevant to the trustor. Social trust means trusting an entity based on information from others (i.e., social evidence). This information could be testimonies from individual witnesses or from a reputation agency, regarding the trustee.

An important practical example is the reputation agency of eBay [2003], where, after each transaction, the buyer can rate the seller. The reputation of a user is then defined as an aggregate of the ratings that it has received from others (i.e., the *raters*). Even having a high reputation of this sort is not a guarantee that the user is trustworthy. Many raters avoid giving bad ratings or may mutually rate each other high [Dellarocas, 2000]. That is, the ratings themselves may not reflect the actual experiences of the users.

Even when the ratings are reliable, an individual has no means to trust the raters. The context in which the ratings were given as well as the evaluation of the services could vary by episode and by rater. The credentials of the information sources (witnesses or reputation agencies) are crucial for interpreting this second-hand information correctly. Unless you have established that the raters are trustworthy, their aggregate ranking would not be sufficient to create trust. That is, in order to create trust through second-hand information, the trustworthiness of the information sources must be established as well [Sztompka, 1999].

1.2 Referrals

A powerful way of ensuring that service providers and the information sources that recommend them are trustworthy is by accessing them through *referrals* [Bonnell et al., 1984; Kautz et al., 1997; Singh et al., 2001]. People commonly use referrals in real life to find useful providers, while 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. One widely used example is the domain name system (DNS), where each unsuccessful lookup results in a referral to a server higher in the name space hierarchy. The referrals of DNS are far from capturing the flexibility of the referrals in human dealings. In real life, a party would give a referral based on specific queries or even based on its relationship with the query originators. The quality of the referrals given creates incentives for the querying party to trust (or not to trust) the referrer.

We claim that these flexible referrals are essential for locating trustworthy services, and we propose that referrals form a key organizing principle for open systems. Consumers can help each other find desired service providers by giving referrals to those providers that have been useful for them. When a consumer interacts with a provider, it can judge the quality of the services it receives as well as the quality of the referrals (if any) given for that service. By keeping track of the quality of the services and referrals, each consumer can establish both local and social trust. More importantly, the consumers can adapt to one another. For example, based on its interactions with others, each consumer can autonomously decide who is most useful for its needs. The referrals through which consumers locate providers create a *referral network*, where the nodes of the network are the consumers and the producers. A consumer links to another party only if the other party has been useful either by providing services or by providing referrals.

1.3 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. This is important because traditional client-server architectures have become inadequate to handle the information needs of Web applications. Most of the time, centralized servers do not readily scale to the size of the Web, and the available computing resources of the clients are under-utilized. The asymmetric interaction pattern of the client-server architectures is being replaced by the symmetric interactions of P2P systems, 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 [2001] and Freenet [2001] 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. Whereas current P2P systems are used for simple applications such as file exchange, the true power of the P2P architecture will arise in more general settings, where the peers can be seen as providing services to one another, in open and dynamic settings.

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 have to establish that these specialized peers are indeed trustworthy, e.g., to ensure that their service recommendations are not based on ulterior motives, such as in the paid-placement search engines of today's Internet, or that any ulterior motives are factored in to determine a suitable service.

1.4 Web Structure

Recently, structural properties of the Web have attracted a lot of attention. Based on traces of the Web, Albert *et al.* [1999] predict that the diameter of the Web, the average shortest distance between any two Web pages, is only 19. This result is important because such a short diameter is the first step in showing that the Web is a *small world network*. A network becomes a small world network when the average length between nodes is small, yet clustering among the nodes is high [Watts, 1999]. Showing that the Web is a small world network would have major implications, such as on how easily the Web can be searched.

Others [Broder et al., 2000; Barabási et al., 2000; Adamic and Huberman, 2000] study in-degree distribution of the Web and come up with stochastic models that can generate these distributions. The main variable of these models is the in-degree of a Web page.

These models generally build on the idea that already popular pages are more likely to attract newcomers than pages with low in-degree. This *preferential attachment* of links explains the observation on the Web of the "rich gets richer" phenomenon.

These approaches seek to understand and model the development of the Web without regard to an understanding of why links emerge. Hence, even though these models are useful, none of them capture the *process* through which links are created. By contrast, referrals provide an intuitive way to model the process through which the links among participants emerge.

When the consumers in our model contact others or generate referrals, they do so with respect to specific service requests. The links on the Web are created by individual parties based on evaluation of other pages, where useful pages are discovered by following links from other pages. Thus, the exchange of referrals mimics browsing behavior and reflects the creation of links on the Web.

1.5 Organization and Summary of Contributions

We study a referrals-based architecture for locating services. This architecture has three important properties. One, the participants can model others based on the quality of service that they receive. Two, the participants can use these models to place varying levels of trust in each other. Three, the participants can choose with whom they want to interact. These properties are crucial in modeling open systems that consist of autonomous and heterogeneous entities.

A further advantage of this architecture is that it can easily be extended to capture real-life characteristics of Web services, e.g., relationships between services as applied to trust. Effects of different characteristics of trust, such as previous interactions, local and social evidence, can be studied in this framework.

In this dissertation, our primary focus is on identifying properties of referral networks, especially those related to the quality and structure of referral networks. The identification of such properties is important because these properties can then be exploited to build effective referral systems for real-life applications.

Chapter 2 introduces our referrals-based architecture for service location as well as our experimental setup in detail. It shows how the referrals of parties induce a natural structure on a system. An immediate consequence of this is that major application classes correspond to different structures. Having a general approach enables us to potentially capture key signatures of different applications and exploit additional properties of these applications.

Chapter 3 studies three aspects of service location. One, it shows how communities emerge as a result of referral exchange and how these communities can be used. Then, it compares our approach to recent work on community mining, which involves centrally collecting and reasoning about links to infer the most useful nodes (conventionally Web sites). Our main result here is that following referrals from trusted parties helps locate desired service providers better than contacting providers identified as authoritative based on others' needs.

Two, it discusses how the quality of the system is influenced by the manner in which referrals are exchanged. We identify the effect of referrals on finding trustworthy service providers in terms of efficiency and effectiveness. Three, it studies structural properties of networks, particularly undesirable network topologies that lead to lower quality networks. The interesting result here is that more referrals help find more answers but the manner in which parties refer and change neighbors can lead to undesirable network structures that disconnect them from others. That is, local actions do not always result in globally useful network structures.

Chapter 4 studies different aspects of the evolution of the structure of referral networks with respect to recent work on Web structure. When parties refer and change neighbors in certain specific ways, based on link structure some agents become considerably more popular as well as more authoritative than others. These asymmetric distributions of popularity and authoritativeness are analogous to those seen on portions of the Web. That is, some characteristic distributions of the Web can be captured through the specifications of agent strategies and environment variables. The fact that such similar distributions can be created, combined with the similarity between referrals and link generation on the Web, suggest that the referrals approach may be used as a predictive model for the Web under different assumptions. Accordingly, this chapter first studies the facts that create these asymmetric distributions, and continues by showing how the system can be used to understand the evolution of a network after perturbation.

Chapter 5 introduces a graph-based representation of services that captures the potential relationships between service types. Using this representation, consumers can propagate their levels of trust in service providers across service types. We first show that such a hierarchical service representation can easily be combined with referrals to exploit the benefits of both. Next, we show that referrals work well with such a hierarchical representation even with limited information, since the consumers use their local information to find a provider for a new service. The chapter continues with a sensitivity analysis of this representation to identify factors that affect its performance.

Chapter 6 discusses the relevant literature with comparisons to our work, describes directions for further research, and outlines how the findings of this dissertation may be used to develop applications of referral systems.

Chapter 2

Technical Framework

We begin with the framework of Singh *et al.* [2001], where a system is modeled as consisting of *principals* who provide and consume services. The principals could be individuals or businesses. Each principal may be offering a different level of trustworthiness in the tasks that it carries out. Each principal is potentially interested in finding trustworthy principals to interact with. Our notion of services is broad, but we discuss two main kinds of services below. These correspond to knowledge management and e-commerce, respectively.

The principals are autonomous. That is, it is not appropriate to assume that a principal will always respond to another principal by providing a service or a referral. When a principal does respond, there are no guarantees about the quality of the service or the suitability of a referral it provides. However, constraints on autonomy, e.g., due to dependencies and obligations for reciprocity, are easily incorporated.

The principals are heterogeneous. The internals of their decision making are not known. That is, the principals may not expose the way they give referrals or the way they evaluate received services. Given the autonomy and heterogeneity of the principals, it is not appropriate to assume that any principal should necessarily be trusted by others: a principal unilaterally decides how to rate another principal.

The above properties of principals can be captured computationally with the notion of *agents*. Agents are persistent computations that can perceive, reason, act, and communicate [Huhns and Singh, 1998a]. Agents can represent different principals and mediate their interactions. That is, principals are seen in the computational environment only through their agents. The agents carry out the book-keeping necessary for a principal to track its ratings of other principals. Moreover, the agents can interact with one another to help their principal find trustworthy parties.

In abstract terms, the principals and agents act in accordance with the following protocol. When a principal desires a service, or when its agent anticipates the need for a service, the agent begins to look for a trustworthy provider for the specified service. The agent queries some of its *neighbors*, which constitute a small subset of the agent's acquaintances that have been found useful in previous interactions. A queried agent may offer to perform the specified service or, based on its *referral policy*, may give referrals to agents of other principals. The referral policy of an agent specifies when the agent should give a referral and whom to refer. The querying agent may accept a service offer, if any, and may pursue referrals, if any.

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 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.

The above framework accommodates the following important properties of open information systems introduced in Chapter 1. One, the agents can be heterogeneous. An agent can offer services or follow policies distinct from all others. Two, each agent operates autonomously based on its local policies. Three, the agents can adapt. Each agent can arbitrarily modify its offerings and their quality, its policies, and its choice of neighbors.

2.1 Applicable Domains

The above framework enables us to represent different application domains naturally. 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 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, 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.

Figure 2.1 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. Consumers are connected to each other as well as to the service providers. These links are essentially paths that lead to service providers. In this model, the service providers are dead ends: they don't have outgoing edges, because

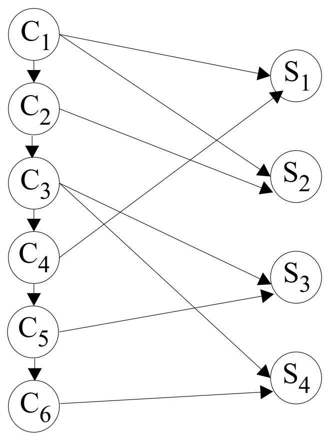

Figure 2.1: A schematic configuration for e-commerce

they neither initiate queries nor give referrals. Thus, their sociability stays low. Their true and modeled expertise may of course be high.

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 their expertise would increase over time. 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 2.2 shows an example configuration of service consumers and providers that corresponds to a knowledge management setting. In this setting, the consumers are not necessarily distinct from the service producers. An agent may be knowledgable in one domain and hence respond to queries regarding that domain. On the other hand, 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

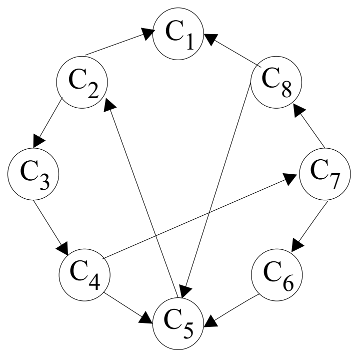

Figure 2.2: A schematic configuration for knowledge management

expertise and sociability.

This dissertation will study the e-commerce domain in depth. The interests and expertise of the agents are represented as term vectors from the vector space model (VSM) [Salton and McGill, 1983], each dimension corresponding to a different domain. That is, each agent has an interest vector and an expertise vector with the same number of dimensions. The interest vector, I_A of an agent A denotes how interested the agent is in different domains. Similarly, the expertise vector, E_A of an agent A denotes how much expertise the agent has in different domains. The value of each element of the vector is between 0 and 1; bigger values correspond to a higher interest or expertise for that domain. The interest and expertise vectors of an agents are not correlated.

2.2 Key Metrics

Similarity. To capture the *similarity* between two agents (based on their interests or expertise), we seek a formula that is commutative, i.e., agent A's interest vector I_A should be as similar to agent B's interest vector I_B as I_B is to I_A. A well-known 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 need to 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. (I_A and I_B are each of length n.)

$$I_A \oplus I_B = \frac{e^{-||I_A - I_B||^2} - e^{-n}}{1 - e^{-n}} \tag{2.1}$$

Capability. The *capability* of an agent for a query measures how similar and how strong the expertise of the agent is for the query [Singh et al., 2001]. Capability resembles cosine similarity but also takes into account the magnitude of the expertise vector. This means that agent that have expertise vectors with greater magnitude are more capable for the given query vector. In Equation 2.2, 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.

$$Q \otimes E = \frac{\sum_{t=1}^{n} (q_t e_t)}{\sqrt{n \sum_{t=1}^{n} {q_t}^2}} \tag{2.2}$$

We introduce and motivate further metrics as needed. The metrics developed here are consolidated in Appendix A.

2.3 Referral Algorithms

Each agent is initialized with the same model for each neighbor; this initial model encourages the agents to both query and generate referrals to their neighbors.

An agent that is generating a query follows Algorithm 1. Since there are no actual principals in the simulations, the generation of queries is automated. More specifically, 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

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
```

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 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. Again, since there are no principals to generate the answers, the answer generation is also done by the agents. If the expertise matches the question, then the answer is the perturbed expertise vector of the answering 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:    Generate referrals to 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, each agent has a chance to choose new neighbors from among its acquaintances based on its neighbor selection policy. The number of neighbors is fixed, so if an agent adds some neighbors it will have to drop some neighbors as well. We tune the simulation so that an agent answers a query only when its expertise matches the query, i.e., it is sure of the answer. This ensures that only the providers answer questions, and the consumers generate referrals to find the providers.

Example 1 Figure 4.8 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. ▮

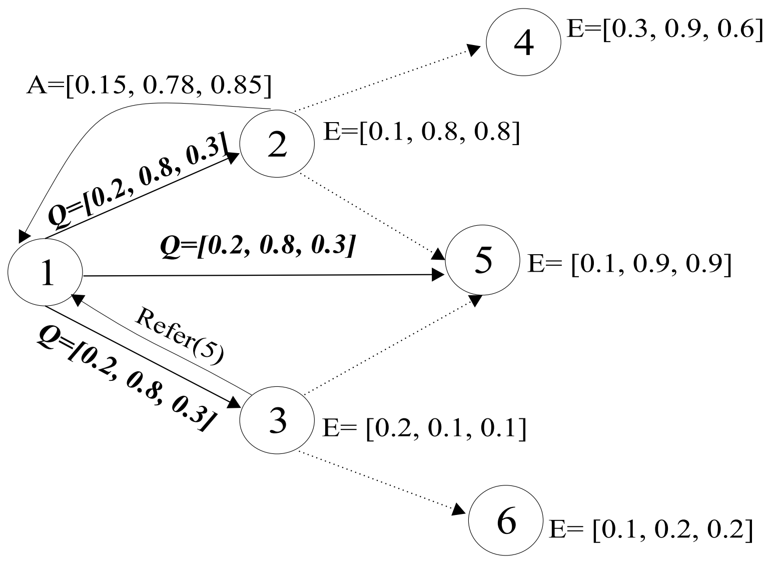

Figure 2.3: An example search through referrals

2.4 Real Applications

Algorithms 1 and 2 are the core of an existing P2P system, called MultiAgent Referral System (MARS) [Yu, 2001]. Each MARS client accepts queries from its user, and follows Algorithm 1 to find answers. Similarly, it handles incoming messages through Algorithm 2, giving referrals to others or letting its user answer a query where applicable. Presently, each user follows default policies for generating referrals and neighbor selection.

In real life applications [Mo, 2001; Yu and Singh, 2002a], the first line of Algorithm 1 1 would correspond to a user request or an agent's anticipation of such a request. Line 13 would directly or indirectly evaluate the answer based on user feedback. Algorithm 2's "generate answer" (line 2) may access a user's documents or, if necessary, interrupt the user.

2.5 Experimental Setup

This dissertation investigates the properties of e-commerce domain via a simulation. Studying the systems 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 like MARS.

The simulations contain 400 agents, where between 5% and 25% of the agents are service providers in one domain, 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 high interest in getting different types of services, but they have low expertise, since they don't offer any services themselves. The interests of the consumers are distributed evenly over the domains, and the interests can span multiple domains.

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 by the capability metric on the query and answer vectors. As discussed above, each agent builds a referral graph to reason about any received referrals. We limit the length of the longest path in referral graphs. That is, a chain of referrals are followed up to this given number of hops, and then dropped. We assume that as the length of the path increases, the expected trustworthiness of the agents decrease.

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, depending on the experiment. Further details of the experimental setup are given in Appendixes B and C.

Chapter 3

Emergence of Communities

Since communities exist in the physical world, it is to be expected that they will emerge in the virtual world as well. On the Web, communities can help us in two ways: (1) identifying interesting and important sites and topics, and (2) fine-tuning the experience of each user by giving us a basis for making recommendations [Zhong et al., 2002]. For these reasons, social network analysis and community mining have garnered much research attention lately.

Communities can be used for two main classes of applications.

- *Endogenous*. The members of a community use the community to find services (including information services). That is, the participants use a community somewhat as people might use their *personal network* to decide what movie to watch or what house to buy. Their personal network involves a part of the social network that is closely related to them. Since the boundaries of communities in real life are amorphous, the participants may not and need not be aware of which specific community they are benefitting from.
- *Exogenous*. The community structure is used to make recommendations. For example, a recommender system might use some features of a community to

which a user belongs to recommend a movie for the user to watch. Conversely, the recommendations might be made to the providers of services so they can fine-tune their offerings for a particular community.

From a sociological standpoint, social network analysis has been one of the most influential methodologies to understand communities in the physical world [Wasserman and Faust, 1994; Scott, 1991]. The underlying idea is to understand social relationships of various kinds among people and to analyze those relationships to determine the communities in which they participate. The relationships between people are a given—they are determined by sociologists, e.g., through ethnographic studies. The graph-based analysis of these communities consider the people as vertices of a graph and the edges as the social relationships observed (e.g., kinship or friendship) between them. The sociological work is not directly applicable in Web-based settings, because the underlying social relationships are not explicit. However, if the underlying relationships can be acquired or inferred, it provides a useful intellectual basis for the computational work.

Influenced by the sociological approach, our referrals-based approach considers interactions among the agents participating in a community. As a graph, its vertices are the agents and the edges represent the neighborhood relation. Good interactions reinforce their social relationships and bring them closer, whereas bad interactions weaken their social relationships. The agents decide with whom to interact. Intuitively, agents will base their decisions on specific feedback or generic policies set by their users, but in terms of its interactions with other agents, each agent is autonomous.

Recently, several approaches have been developed to discern community structures from Web pages [Kumar et al., 1999; Flake et al., 2002]. The Web is viewed as a graph where the vertices are Web pages and the edges are hyperlinks from one page to another. Communities are defined as patterns of self-similarity as in co-citations or highly clustered cohesive blocks. Large corpora of pages can thus be mined centrally to determine graph structures to be interpreted as communities.

3.1 Community Analysis

We first give a conceptual analysis of link-based definitions and argue that these approaches are not sufficient to describe real communities. Then, we provide a quantitative evaluation and show that in fact it is not even necessary to mine communities to benefit from them.

Link-based definitions of communities use a form of co-citation. Perhaps the best known of the existing approaches is that of Kumar *et al.* [1999]. This is reminiscent of hubs and authorities in Hyperlink Induced Topic Search (HITS), a well-known Web page ranking technique [Kleinberg, 1999]. The HITS algorithm assigns an authority value and a hubness value to each node in the Web graph. A node has high authoritativeness if nodes with high hubness values point to it. Similarly, a node has a high hubness value if it points to authoritative nodes. Reflecting a similar intuition, Kumar *et al.* define communities in terms of related sets of *fans*, which ideally point at lots of centers, and *centers*, which are ideally pointed to by lots of fans. Kumar *et al.* propose that any community structure should contain a bipartite core where the fans and centers constitute the two independent sets. If all n fans point to a set of m center pages, then they are likely to share a common topic and therefore be a community. Especially in the case of high n and m, the likelihood of being a community is assumed to be higher. In addition, all other nodes that are pointed by the fans, and all the nodes that point to at least two centers are added. In their experiments, Kumar *et al.* use (3,3) bipartite cores. For example, in the graph in Figure 3.1, the nodes 1 to 8 could denote a community, where the nodes 1 to 3 are the fans, and the nodes 4 to 6 are the centers. These nodes constitute the bipartite core (shown in solid lines). The node 7 is added because it points to two centers (5 and 6) and the node 8 is added because it is pointed to by a fan (1). These expansions are shown in dotted lines. Even though there is a link between nodes 6 and 9, node 9 is not added to the community. We refer to these communities as bipartite communities.

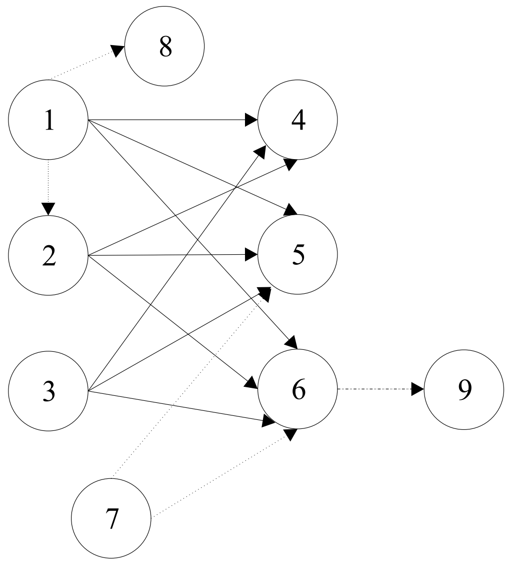

Figure 3.1: Nodes 1 to 8 form a bipartite community

There are four major limitations of link analysis as a basis for understanding communities:

- A link does not necessarily correspond to a social relationship. A definition of communities based on links assumes that two pages have a "social" relationship if one links to the other page. However, the connection between a link on one's page and a social relationship is tenuous at best, especially because the semantics of the links between the nodes is not defined. For this reason, although graphs can be extracted, it is not automatically obvious that these graphs correspond to communities as we would intuitively consider them.

  With reference to Figure 3.1, consider three consumers (nodes 1 to 3) interested in three distinct domains. Each one of them chooses the same three service providers (nodes 4 to 6), each of whom provides a service in one domain. These six nodes form a bipartite core since all three consumers point at all three providers. It is not obvious why this structure would denote a community, when the three service providers are not even providing the same service.

- The structures may be interpreted differently in different settings. For example, consider the two application domains discussed in Section 2.1. An independent set of service providers is common in e-commerce: the service providers are not looking for services themselves, so they do not point to each other. In knowledge management, on the other hand, every agent is potentially looking for services (i.e., knowledge in this case). In other words, having an independent set of agents can have several implications, such as different evaluation of services, being unaware of each other, and so on. Hence, the structures alone may not be sufficient to accurately represent communities.
- Co-citation as a relationship seems to be almost incidental among the parties so related, whereas one would expect socially related parties to locate each other, potentially by traversing the network. With co-citation the participants are not aware of each other. For example, nodes 2 and 3 may not know they are in the same community. That is, node 2 cannot locate node 3 to send its queries. Hence, communities based on co-citation cannot be used for endogenous applications.
- Conceptually, communities are discovered in a central manner. Each node has to expose all of its links and a central server has to put together a graph to later mine the communities. However, this indicates a grave risk of violating the privacy of the potential participants, because some nodes would want to

keep some of its links private. Clearly, mining can work best for only static Web pages, which the participants have made available publicly. Each link is interpreted as an endorsement. However, this approach fails to apply when participants decide not to reveal their endorsements publicly. Conversely, the beneficiary of any recommendation may prefer they come from parties it knows.

There are three major advantages of the referrals approach.

1. Agents can provide semantics to the edges in the graph. In other words, because agents maintain models of others, they are able to annotate their links to other agents in terms of those models. For example, an agent i may believe that agent j is the best source for information on travel and agent k is the best source for information on cooking.

2. Unlike link-based communities, two agents are part of the same community because of their interactions with each other, rather than their interactions with others. Agents are aware of whom they are dealing with, and they choose whose service recommendations they will take.

3. No formal community needs to be identified for an agent to function correctly. Communities emerge around each agent. Each agent automatically exploits them and evolves them as it goes about its business. However, for link-based approaches, first the link structure needs to be mined by a central entity and communities have to be identified. Only then may the communities be used. Unlike link-based approaches, in our referral-based approach no central authority need know what the communities are.

However, in order to perform our analysis, we mine the communities. Doing so enables us to compare our approach to the link analysis approach.

3.1.1 Methodology

Comparing referrals approaches with link analysis in quantitative terms is potentially tricky, because link analysis is applied on static Web pages, whereas referrals apply between agents providing and seeking services. There is no widespread practical deployment of such service location schemes. However, a comparison is possible when we consider how links are created. The creation of links on the Web is based on micro evaluations and decisions by independent players. This process of neighbor selection is mimicked well by adaptive referrals.

In mining communities, two properties are worth considering.

- Communities may not have clear-cut boundaries. Previous approaches view communities as crisp structures in that an agent is either a member of a community or not. On the other hand, a community may have many members who differ in their level of membership in the community. Accordingly, our approach is based on ranking members of a community based on their level of membership. An agent may belong to several communities in varying degrees.
- Strength of the links matter. For instance, in some cases, to conclude that the agent is part of a community, it might be enough to show that it has one strong link to a member of a community, whereas if the agent has weaker links to community members, more links might be required.

We consider mining communities of service consumers for different domains. As an example, consider the travel domain. There may be several travel agents represented as service providers. Some service consumers are interested in finding travel agents and query other service consumers to locate the providers. The service consumers who help find the travel agents are found to be sociable by the travelers, since the sociable agents' referrals help in locating the providers. Since sociable agents are more influential in locating the service providers, we use sociability of the agents to rank their involvement in the community. A consumer belongs more strongly to a community if more consumers find him to be sociable. Note that sociability is subjective. For instance, agent A may view agent B as sociable, whereas a third agent may not. When this is the case, the agent who is part of the community has a bigger say. In other words, if A is part of the community, it can judge B's contribution better than someone outside the community, or someone less involved in the community. In this regard, members of the community decide who should be in the community [Yolum and Singh, 2003b]. This recursive definition is inspired by the PageRank algorithm.

PageRank. PageRank is a metric used by Google to rank Web pages that are returned for a query [Brin and Page, 1998]. 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 3.1, where $P(i)$ denotes the PageRank of agent i, K_i denotes agents that have i as a neighbor, and N_j denotes the set of agents that are neighbors of j. In addition to accumulating PageRanks from incoming edges, each node is assumed to get a minimum PageRank of $(1 - d)$, where d is taken to be 0.85 as in the original paper [Brin and Page, 1998].

$$P(i) = d \sum_{j \in K_i} \frac{P(j)}{|N_j|} + (1 - d) \tag{3.1}$$

Referral Communities. The PageRank calculations for the Web are performed on a directed unlabeled graph. Here, we build on this idea to mine communities. As mentioned above, the neighborhood relations among the agents induce a directed graph, where each node denotes an agent. An edge (u, v) exists if agent u values agent v's expertise, sociability, or both. This valued expertise or sociability may be in one or more domains. We extend the representation as follows. First, the graph structure is enhanced by adding labels to the edges of the graph, where the label on an edge (u, v) denotes v's sociability from u's point of view (in our notation, this is $\sigma_{u,v}$) for one domain. u may model v's sociability for different domains. In other words, u might find v sociable for one domain, but not sociable for many other domains. Second, the sociability rank for each agent is calculated per domain as given in Equation 3.2. Below, $S(i)$ denotes the sociability rank of agent i, K_i denotes agents that have i as a neighbor, N_j denotes neighbors of j, $\sigma_{j,k}$ denotes the sociability of k for j.

$$S(i) = d \sum_{j \in K_i} (S(j) * \frac{\sigma_{j,i}}{\sum_{k \in N_j} \sigma_{j,k}}) + (1 - d) \qquad (3.2)$$

In PageRank calculations, at each iteration, each node distributes its PageRank to its neighbors equally. Here, on the other hand, each node distributes its sociability rank based on the sociability weights on the edges.

Example 2 Consider an agent j with neighbors i, k, and l such that $\sigma_{j,i} = 0.8$, $\sigma_{j,k} = 0.2$, and $\sigma_{j,l} = 0.2$. Then, j will contribute $\frac{0.8}{0.8+0.2+0.2} = 0.67$ to i's sociability rank, whereas only $\frac{0.2}{0.8+0.2+0.2} = 0.17$ to k and l each. ▮

The above definition of communities captures two important notions. One, members of the communities implicitly decide on the other members. Two, the members are chosen based on how helpful they have been to others. This implies that an agent may belong to a community more strongly than a second agent even though both agents have the same neighbors.

Example 3 Figure 3.2 shows part of a referral network, where agents 1 and 2 are pointing at the same set of agents 3, 4, and 5. However, agent 1 is being pointed to by more agents

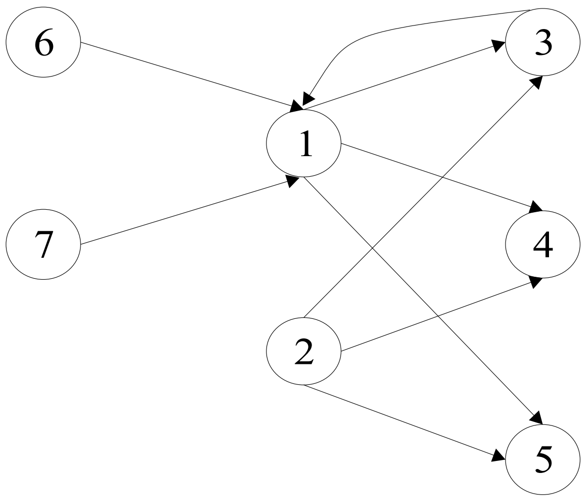

Figure 3.2: Agent 1 is ranked higher than agent 2

because it has been found sociable by more agents. Therefore, the ranking of agent 1 is higher than that of agent 2, which denotes that agent 1 belongs more strongly to the community than agent 2. ▮

3.1.2 Metrics

We define the quality of a network to be the ease with which agents find useful service providers. We study two variations of quality: direct quality and nth best quality. Both metrics calculate the quality from a single agent's point. The quality of the network is then defined as the average over all agents.

Direct Quality. The *direct quality* viewed by an agent reflects, via Equation 2.2, 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.

Quality. 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 are from the agent, the less their contribution to the quality seen by the agent. The contribution of agent B to agent A's quality is given by:

$$\frac{I_A \otimes E_B}{path(A, B)} \tag{3.3}$$

where the $path(A, B)$ is the shortest path length between agents A and B.

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 via its nth 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 quality of the agent's neighbors 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 the agent has.

3.1.3 Results

We evaluate our approach by comparing it to bipartite communities. To generate bipartite communities, we find bipartite cores of size (6,3). We expand each core by adding all the nodes pointed to by the fans and all the nodes that point to at least two centers. Then, we run the HITS algorithm (mentioned in Section 3.1) to find the authorities and hubs. A node has high authoritativeness if nodes with high hubness values point to it. Similarly, a node has a high hubness value if it points to many authoritative nodes. Since the communities are targeted for locating services, the nodes with high hubness values are expected to be most useful to others. For this reason, we use the hubness values to rank the nodes of the community.

Correlation First, we calculate the correlation between bipartite and referral communities using Spearman correlation, given in Equation 3.4 [Kendall, 1945].

A correlation value of 1 shows that the members of the communities are ranked the same in both approaches, whereas a correlation value of -1 shows that the members of the two communities are ranked in reverse order. Correlation values around 0 denote that the rankings are not correlated.

Below, C and D denote two communities, C_i and D_i denote the rank of agent i in communities C and D, respectively, and n denotes the size of the communities. When comparing the two communities, the community size, n is taken to be the size of the bipartite community found. The top n agents from our ranking are then taken for comparison.

$$\rho(C, D) = 1 - \frac{6\sum_{i=1}^{n}(C_i - D_i)^2}{n(n^2 - 1)} \tag{3.4}$$

We choose 10 communities for comparison. The choice for the communities is arbitrary, except that the chosen communities vary in their size, where the smallest community has 32 members and the largest community has 238. The average correlation among the communities is -0.65, with the correlation values varying from -0.3 to -0.9. The fact that there is no positive correlation between the communities means that the rankings of the two communities do not agree. Based on preliminary studies on the distribution of the correlations, we conjecture that as the size of the communities increase, the ranking of the communities become less correlated; i.e., the absolute value of $\rho(C, D)$ approaches 0.

Quality The community of service consumers for a service should be able to locate the service providers easily. The quality metric captures this intuition. Below, we compare referral and bipartite communities in in terms of their total quality.

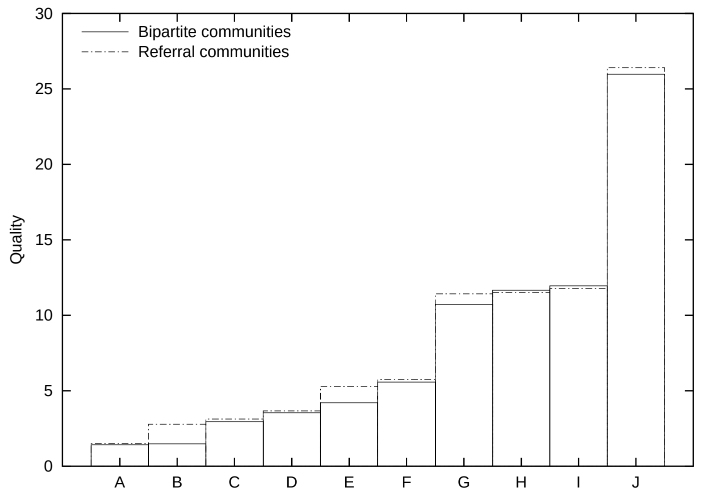

Figure 3.3: Quality comparison of bipartite and referral communities

Figure 3.3 gives a histogram of this comparison. The X axis shows the communities, labeled with letters. The Y axis shows the total quality of the communities. The solid lines denote bipartite communities and the dashed lines denote referral communities. Eight of the referral communities outperform bipartite communities in their quality. Only referral communities H and I receive a slightly worse quality than the corresponding bipartite community.

Finding higher quality communities is especially important for exogenous applications [Domingos and Richardson, 2001]. For example, if a service provider is promoting a new product, the set of customers that are likely to use it are the ones that can actually locate the provider in the first place. Hence, the new product should be targeted to the community that yields the higher quality in terms of locating providers.

One-size doesn't fit all We study the authorities of a community in terms of how well they serve the query needs of the community members. On one side, for a bipartite community, we rank the members based on their authority from the HITS algorithm. From the community, we make five agents generate queries and pose them to the top four authorities. On the other side, we make the agents look for answers to the same queries through a referral process as shows in Figure 3.4.

Consider the community of Figure 3.4. After running HITS, agents 4 and 5 are found to be authorities. In the case for bipartite communities, agent 1 generates a query and poses it to 4 and 5. For the referral communities, it poses the query to its choice of neighbors, in this case to agent 6 who gives a referral to agent 9 (bold lines).

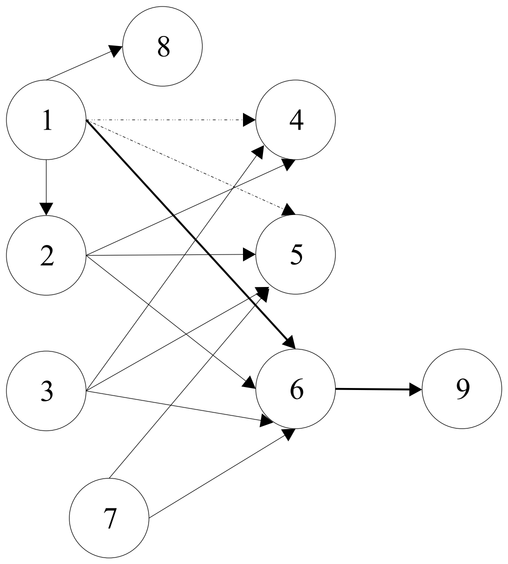

Figure 3.4: Referral process based on the community of Figure 3.1

Figure 3.5 plots the number of good answers for each agent. Four of the five agents get more good answers by following referrals, rather than by posing their query to the authorities. The last agent gets an equal number of good answers with both approaches.

Observation 1 Authorities that are optimized for everybody's needs are not always effective for individual needs. On the other hand, when agents follow referrals from their personal network, they can find a greater number of useful answers.

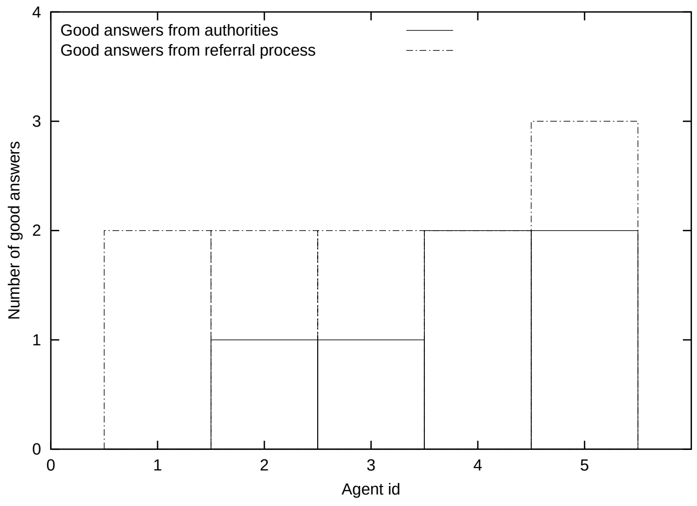

Figure 3.5: Comparison of good answers

3.2 Performance Evaluation

Naturally, the performance of the referral approach is affected by the individual policies used by the agents. Our focus is not on finding optimal strategies for the agents. Instead, we are interested in the emergent global properties of the system. The properties we emphasize are the structure of the networks and their effectiveness. For both cases, we evaluate the influence of these properties on the ability to locate services with some simplistic referral and neighbor selection policies.

We study how the referrals affect the quality of the systems using three referral policies that vary primarily in their level of selectivity [Yolum and Singh, 2003a].

1. *Refer all neighbors.* Agents refer all of their neighbors; agents are not selective about whom they refer. This is a special case of the matching policy with the capability threshold set very low. This resembles Gnutella's search process where each node forwards an incoming query to all of its neighbors if it doesn't already have the requested file [Kan, 2001].

2. *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. By increasing the threshold, the selectivity of the referrals is increased as well.

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 the agent that is most likely to have the requested information [Langley, 2001].

3.2.1 Effectiveness

The effectiveness of a system measures how easily agents find useful providers. We measure effectiveness using the nth best quality metric. Intuitively, the way agents give referrals influences the quality of the system because referrals are the only way to discover new service providers.

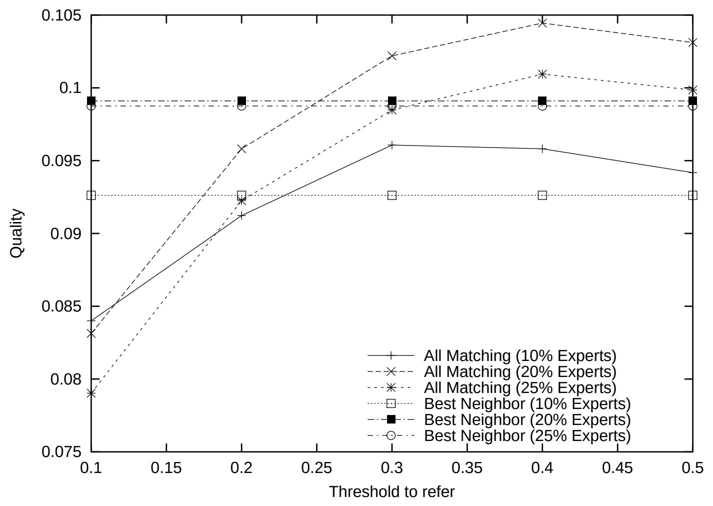

Figure 3.6: Effectiveness of referral policies

We study the performance of different policies by varying the capability threshold. Figure 3.6 plots this threshold versus the quality of the network for different policies. In Figure 3.6, 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.

Among the three policies, *Refer all* performs the worst for all three populations. As seen in Figure 3.6, 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, where the best threshold increases with the percentage of experts in the society.

Observation 2 Exchanging more referrals does not guarantee that the quality of the network will be high. The structure of the network can prevent consumers from locating some of the service providers.

3.2.2 Efficiency

Each agent in the referral network is autonomous and may well have unique 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 2.1 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 correct answers received to the number of agents contacted. Figure 3.7 plots the capability threshold versus the efficiency for different referral policies.

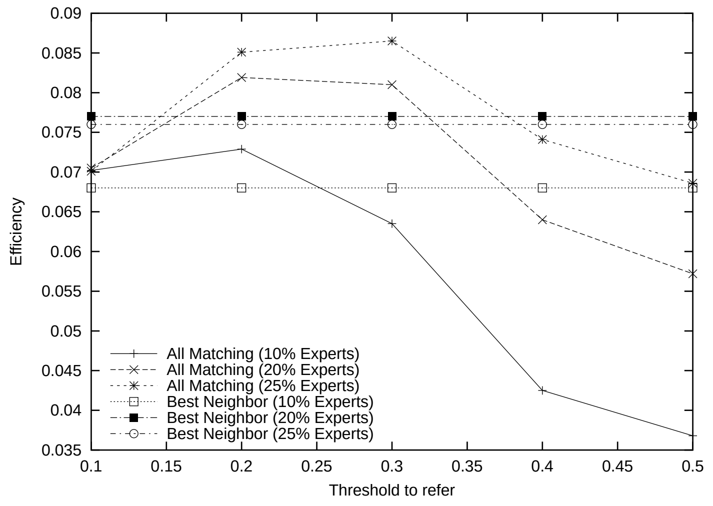

Figure 3.7: Effect of selectivity on efficiency

Refer all matching with high thresholds (e.g., 0.4, 0.5) have the least efficiency. Since these policies are selective, few referrals are given. Hence, most of the time, the agents cannot find good answers, reducing the overall efficiency. On the other hand, 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 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*.

Observation 3 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.

We analyze the combined results on effectiveness and efficiency in three cases. First, with higher thresholds of *Refer all matching* policy, the agents can potentially reach the experts, but since referrals are given highly selectively, most of the time they cannot get referrals to locate the experts and pose their queries. Second, with smaller thresholds of *Refer all matching*, not only can the agents reach the experts, but since the referrals are less selective, they can locate the experts and get good answers. This is also the case for the *Best neighbor* policy, 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*), we would expect agents to be able to locate experts better and get closer to them. If the agents get good answers, then they are finding the experts, yet their ability to reach the experts (measured in quality) is still lower than all other policies. The reason for this is that the agents are close to a few experts which ensures that they get good answers. On the other hand, they are isolated from many other useful providers. That is, the structure of the referral network may evolve in ways that isolate some of the experts from the agents. The next section studies these possible undesirable structures in greater depth.

3.3 Structure

At certain intervals during the simulation, each agent has 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 referral network.

What would happen if each agent chose the best service providers as neighbors? Or would it be 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 network 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 by which an agent selects the best m of its acquaintances to become its neighbors. W denotes the weight of the sociability in choosing a neighbor.

- *Weighted average.* Sort acquaintances in terms of a weighted average of sociability and how their expertise matches the agent's interests.
- *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 structure of the network. The emerging network structures may sometimes be undesirable, mainly because they may cause some agents to be unreachable by others. Essentially, this can happen even when the agents are each locally taking the best action. In other words, an agent's neighbors could be a good match for the agent, yielding high direct quality. But, if the agent cannot reach others, the quality of this agent as well as the quality of the network will still be low.

Consider an extreme case, where each agent has only one neighbor. Agent i and agent j establish a reciprocal relation. Although each of them could be a good match for the 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 network will go down. Figure 3.8 plots the direct quality metric versus the quality metric for different neighbor selection policies. When the direct quality of the network increases, the quality does not necessarily increase.

Observation 4 High direct quality does not guarantee a high global quality. That is, having useful neighbors does not guarantee that agents can locate all useful service providers.

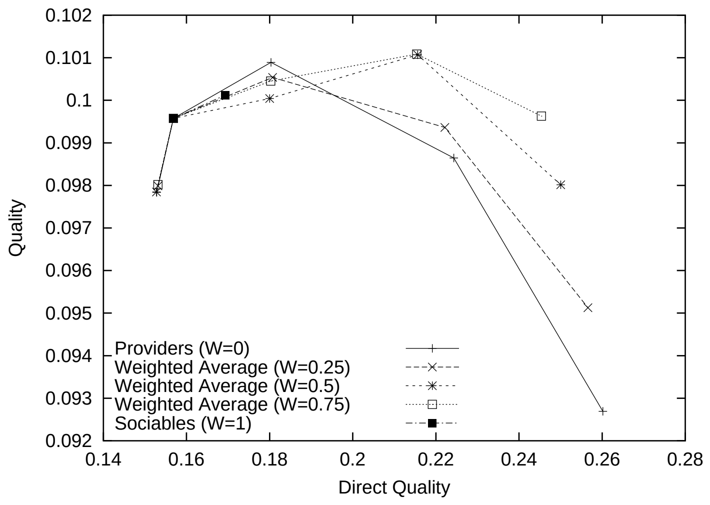

Figure 3.8: High direct quality does not guarantee high quality

3.3.1 Bipartite Graphs

Consider a bipartite graph. A graph G is bipartite if it consists of two independent sets, i.e., two sets of pairwise non-adjacent 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 to 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.

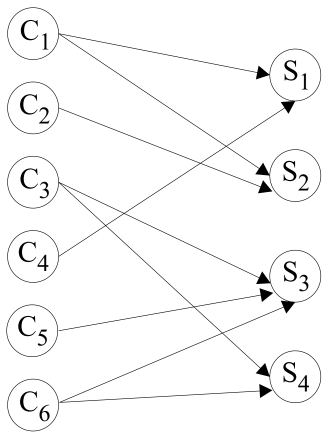

Figure 3.9: A configuration where consumers cannot exchange referrals

We observe that 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 of the same population that allow reachability to other 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 a few edges were taken out from the graph. 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 the how close the graph is to becoming bipartite. In general, detecting if a graph is bipartite is easy. On the other hand, determining the number of edges that it varies from a bipartite graph is NP-complete [Garey and Johnson, 1979]. Here, however, the semantics of the nodes serve 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.

Figure 3.10 plots η after every two neighbor changes for *Providers* policy for a particular run. After each neighbor change, the number of consumers that point to other consumers drop, since each agent finds useful providers to point to. Eventually, the number of edges between the consumers is only 1.

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 the *Providers* policy, the same effect does not hold for *Weighted Average* or *Sociables* policies, since with these policies consumers may choose some other highly sociable consumers as neighbors.

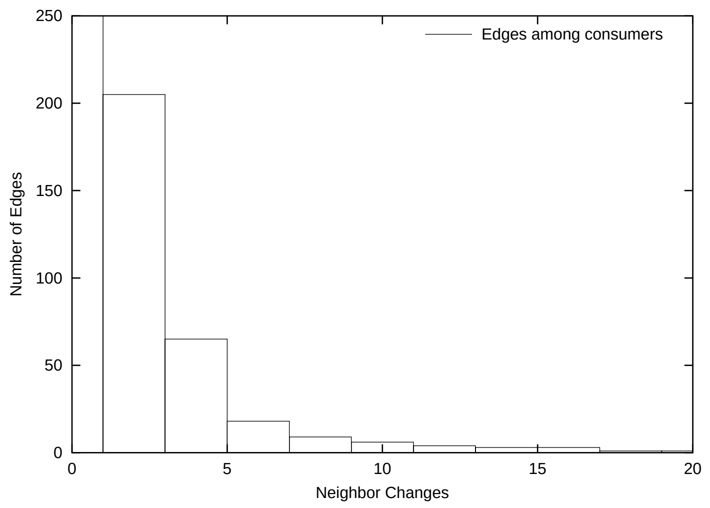

Figure 3.10: The number of edges among consumers decreases

Observation 5 When choosing neighbors, if agents prefer only expertise (use *Providers* policy), then the network can evolve into a bipartite graph, which prevents the consumers from exchanging referrals.

3.3.2 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 [West, 2001]. That is, 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 who will not be able to find at least one service provider. Since the consumers are the only sociable agents, consumers who will choose to be neighbors with other consumers only. In the worst case, this results in the providers being totally isolated from the consumers. We observe that in a population where each agent exercises the *Sociables* policy, the graph ends up with more than one weakly-connected component, as shown in Figure 3.11.

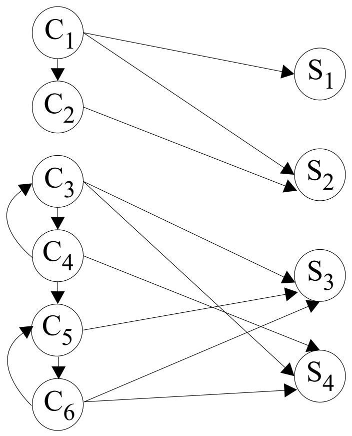

Figure 3.11: A configuration where some consumers cannot reach some providers

Observation 6 When agents use the *Sociables* policy, the network can become disconnected. This may prevent the consumers from locating some of the service providers.

3.3.3 Clustering

Watts defines the cliquishness of a graph as the likelihood of the neighbors of an agent being neighbors with each other [1999]. The cliquishness coefficient for each agent i measures the ratio of actual edges among its neighbors to the possible edges among the neighbors, as shown in Equation 3.5. Below, N_i denotes the set consisting of node i's neighbors. M_i denotes all the edges between the nodes in N_i.

$$\nu(i) = \frac{|M_i|}{|N_i|(|N_i| - 1)} \tag{3.5}$$

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

Interest clustering denotes how similar the neighbors of an agent are in terms of their interests. We use an interest clustering coefficient, similar in motivation to Watts' cliquishness coefficient. Instead of counting edges between the nodes, we use Equation 2.1 to measure how similar the interests of two agents are.

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 3.6, V_i denotes the set of agents that contain i and i's neighbors, and M_i denotes edges between nodes in V_i.

$$\gamma(i) = \frac{\sum_{(u,v)\in M_i} I_u \oplus I_v}{|V_i|(|V_i| - 1)} \tag{3.6}$$

Figure 3.12 plots the interest clustering after every two neighbor changes. Each plot corresponds to a different neighbor selection policy. The interest clustering of the graph

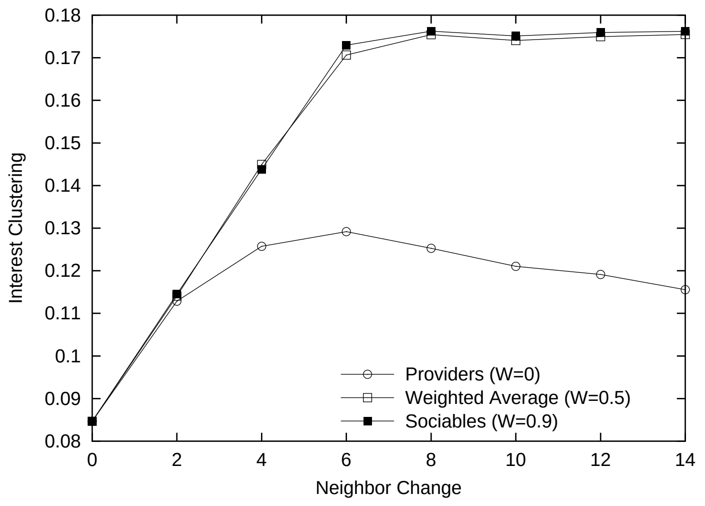

Figure 3.12: Increase in interest clustering over neighbor changes

increases when the agents put greater emphasis on sociability when choosing neighbors.

Observation 7 When agents value expertise more (follow the *Providers* policy), the interest clustering is small since (1) there are few edges between the experts and (2) the interests of the consumers are not necessarily similar to the interests of the experts. On the other hand, when sociability is valued more, 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 thus be considered sociable.

Next, we study the correlation between interest clustering and quality. Figure 3.13 plots the quality of the network for different values of interest clustering. Each plot corresponds to a different neighbor selection policy. The data are plotted after every second neighbor change.

We observe that interest clustering decreases with an increase in quality. An increase in

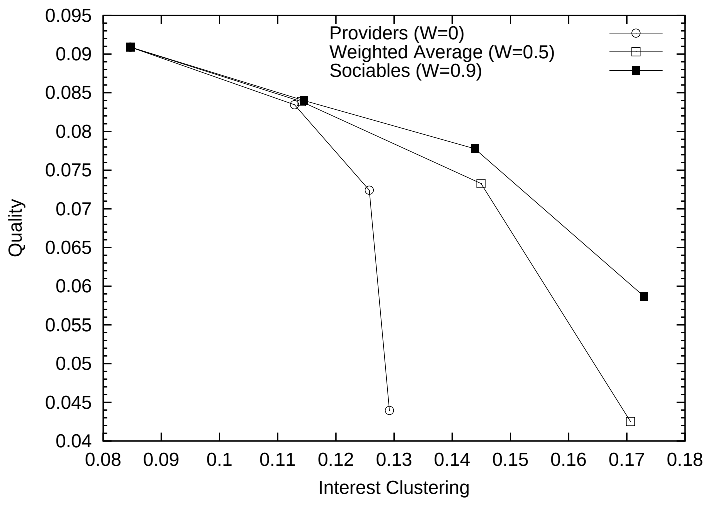

Figure 3.13: Quality versus interest clustering

quality indicates that some consumers are getting closer to the capable service providers. This decreases the 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 qualified travel agent. As soon as one of them discovers a qualified agent, the quality of the network will increase. Further, when requested, it will refer this new travel agent to its neighbors. The neighbors may eventually point to the travel agent instead of another traveler. This will decrease the interest clustering of that particular group of travelers.

Observation 8 Becoming neighbors with agents with similar interests does not guarantee finding useful service providers.

Interest clustering has an interesting consequence: congestion. Informally, an agent in the network is *congested* when its in-degree is significantly higher than others. The idea of congestion here is similar to the one in computer networks. The main difference is that in computer networks the congestion is measured by the difference of out-degree and in-degree. If there are more packets coming into a node than are leaving the node, then the node will be congested [Tanenbaum, 1996]. 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 3.14, 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 greater than 20. The remaining agents have an in-degree below 10 and hence not shown in the graph. Contrast this with 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. We observe that increasing interest clustering increases congestion.

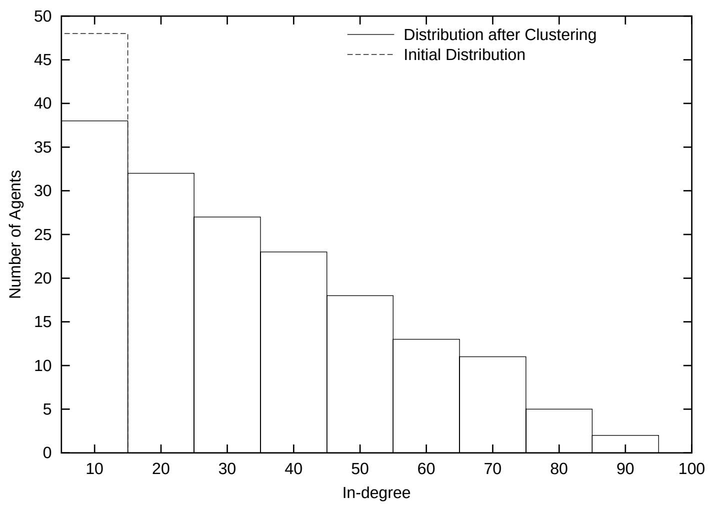

Figure 3.14: Distribution of in-degree before and after clustering

Naturally, while some nodes are getting high in-degree, others have remarkably low in-degree. This raises questions about the in-degree distribution. What are the factors that affect in-degree distribution? How do the authorities emerge? We study these questions in the Chapter 4.

Chapter 4

Emergence of Authorities

Two well-known algorithms for identifying authorities on the Web, PageRank and HITS, have been introduced in Chapter 3. These algorithms assume that links between pages indicate some sort of an endorsement. This assumption leads to the heuristic that a metric such as the PageRank of a page measures the page's authoritativeness. That is, 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. After the Web is mined to generate a graph, these algorithms are used to find authorities.

Even though identification of authorities is useful, these approaches do not provide a view of why and how authorities emerge. That is, the process that creates the authorities has not been studied. Current stochastic models of the Web (e.g, [Albert et al., 1999; Pennock et al., 2002]) are geared toward capturing the growth of the Web in terms of evolving in-degree distributions. But, since they do not capture the reasoning by each creator of a link, other emerging notions, such as authoritativeness or trust, cannot be captured.

Our referrals-based approach for finding trustworthy service providers gives us a direct means to model the evolution of a social network. We can consider the various representations and strategies or policies used by the agents along with distributions of true authoritativeness and relate these to each other. Our study of the processes of linkage and referrals gives us a basis for understanding how authoritativeness emerges. In addition to its theoretical value, understanding the process has two major consequences. One, the process can be used to predict the evolution of the system. Two, better mechanisms can be designed for identifying and benefitting from authorities.

The remainder of this chapter is organized as follows. Section 4.1 analyzes different in-degree distributions that result from different agent policies. Section 4.2 studies factors that affect the distribution of PageRanks. Section 4.3 studies the ways PageRank distributions can evolve after perturbing the population, based on neighbor selection policies.

4.1 Evolution of In-degree Distributions

We perform a series of experiments to determine the in-degree distribution of the graph as induced by neighbor relations. Initially, each agent has an in-degree between 0 and 10. By exchanging referrals, agents learn about experts and sociable agents. Hence, after each neighbor change, the in-degree of experts and sociables increases, whereas the in-degree of the less qualified agents decreases. In populations where the number of qualified agents is low, this distribution of in-degree would imply that the few qualified agents will have high in-degree and the remaining non-qualified agents will have low in-degree, if any. This type of distribution suggests a *power law* distribution.

A network exhibits a power law distribution if the *rank* of each node is proportional to a power of its in-degree. The ranks are assigned based on the in-degree of the nodes; the node that has the highest in-degree gets the first rank, the node with the second highest in-degree gets the second rank, and so on. The nodes with equal in-degree are listed arbitrarily.

Equation 4.1 gives the generic form of the power law distribution, where i denotes a node, $P(i)$ denotes the in-degree of i, $R(i)$ denotes the rank of node i, κ is a constant, and α is the power law exponent.

$$R(i) = \kappa P(i)^{\alpha} \tag{4.1}$$

The power law distribution of two variables captures the linear relation between the logarithms of these two variables such that α denotes the slope of the line and $log\ \kappa$ denotes the y-intercept (Equation 4.2).

$$\log R(i) = \log \kappa + \alpha \log P(i) \tag{4.2}$$

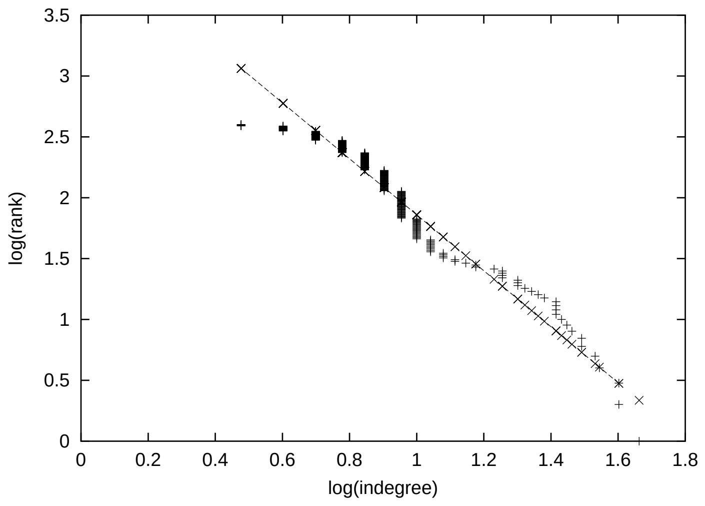

Figure 4.1: Log(in-degree) versus log(rank)

Power law distributions are shown to hold for distribution of business firm sizes [Axtell, 2001], the city populations in the U.S. [Marsili and Zhang, 1998], distribution of words in natural languages [Zipf, 1949], as well as distributions of in-degree and out-degree of the Internet [Faloutsos et al., 1999] and the Web [Albert et al., 1999; Huberman, 2001]. Recently, Pandurangan *et al.* [2002] showed that the distribution of the PageRanks on the Web follows a power law as well.

Figure 4.1 shows a plot of data points and the line fit for a population which has 10% service providers and 90% service consumers, where the service consumers are distributed equally over possible domains. The data set contains the log of the in-degree and the log of the rank of 400 agents. The least squares method is used to find the line that best fits these data points. As is customary, the square of the correlation coefficient (R^2) is used to decide whether a population obeys a power law distribution. The R^2 denotes how well the estimated line fits to the data [Walpole and Myers, 1978].

We first use our model to study the in-degree distributions. We consider the evolution of in-degrees under the *Providers* and *Sociables* neighbor selection policies. *Weighted Average* policy (with W=0.5) is similar to the *Sociables* policy, so we do not consider it as a separate case.

4.1.1 Providers

When agents use the *Providers* neighbor selection policy, each consumer replaces its initial neighbors that are consumers with any useful providers it finds. Thus, there is a rapid increase in the in-degree of providers. The providers that are located first are propagated through the system, accumulating higher and higher in-degree. Meanwhile, many agents still have low in-degree, either because they do not have any expertise to serve others or they are providers waiting to be discovered. Power law distribution of in-degree emerges when some but not all consumers have found the providers. The few providers that are discovered earliest have high in-degree, while the remaining providers have lower in-degree because they are in the process of being discovered, and many consumers have low in-degree.

If the referring policy encourages referrals (e.g., *Refer all*), then all the consumers locate the providers after a few queries. This is analogous to the case when the graph turns into a bipartite graph, as was discussed in Section 3.3.1. After the graph turns into a bipartite

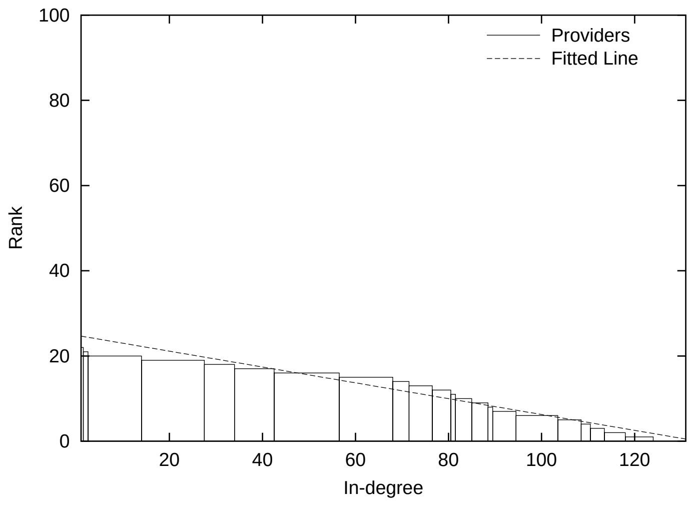

Figure 4.2: Linear relation between in-degree and rank

graph, the in-degree distribution no longer follows a power law. At this stage, almost all consumers have in-degree zero, and the possible in-degrees are shared among the providers with a linear distribution, as shown in Figure 4.2.

Observation 9 When agents prefer expertise and exchange many referrals, each consumer locates as many providers as possible. Hence, the possible in-degree is shared only among the providers.

On the other hand, if the referral policy prescribes selective referring, then some of the consumers do not find as many providers as their allowed number of neighbors. They continue to point at their original neighbors. When this is the case, then the power law distribution remains (Figure 4.3).

Observation 10 When agents prefer expertise but do not exchange many referrals, only a few providers are located by most consumers. Most consumers and most providers have low in-degree, while only a few providers have high in-degree.

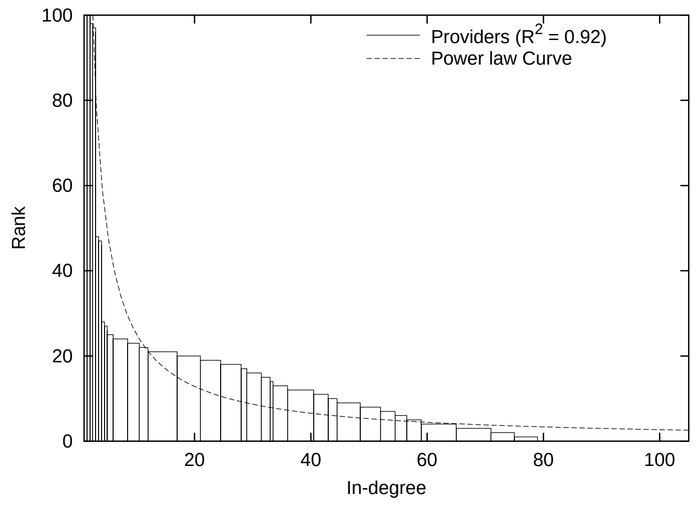

Figure 4.3: Power law distribution of in-degree for *Providers* policy with selective referrals

4.1.2 Sociables

When agents follow the *Sociables* policy, providers as well as sociable consumers that point at the providers increase their in-degree. The consumers that initially point at experts have an advantage; they will be considered sociable by the agents that point to them early on. After a few queries, these agents receive high in-degree right away due to their sociability. Many other consumers do not point to providers at the beginning of the simulation, and hence are not considered sociable.

$$\begin{tikzcd} C_1 \arrow[r] & C_2 \arrow[r] & C_3 \arrow[r] & S_4 \end{tikzcd}$$

Figure 4.4: An example network

The agents that initially point at the providers gain the most in-degree, since they are recognized as sociables, and are being pointed to by more and more agents. The consumers that learn about the providers after several queries are not considered to be as social as the initial sociables. Here again, the distribution moves from a power law distribution toward a linear distribution, where the in-degree is shared among initially-sociable agents. Figure 4.5 shows the distribution when the referrals are not selective.

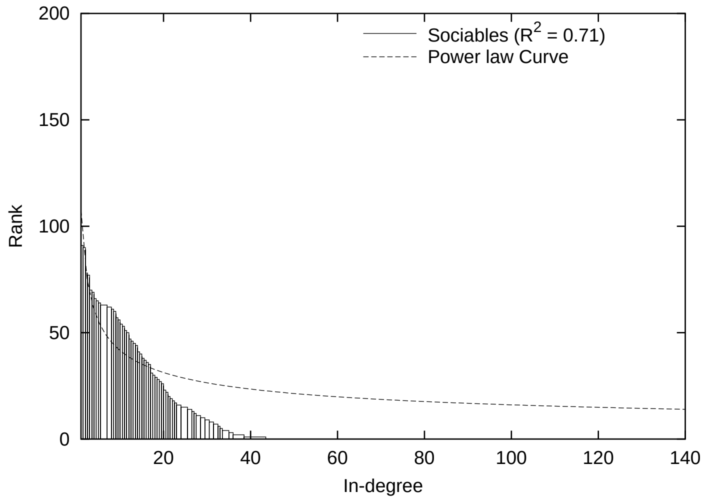

Figure 4.5: In-degree distribution for *Sociables* policy with non-selective referrals

For example, let agent C_1 point to agent C_2, a consumer who does not initially point to any of the providers, as in Figure 4.4. If the referrals are not selective (e.g., *Refer all*), when agent C_1 sends a query to agent C_2, agent C_2 will refer all of its neighbors, who in turn will refer all of their neighbors, until the maximum referral graph length is reached. When agent C_1 finds a provider, the agent C_3 who gives the final referral to the found provider S_4 will get the highest increment in terms of sociability. Agent C_1 will replace agent C_2 with agent C_3, since C_3 has been more useful. Hence, with *Refer all* policy, agents who are initially sociable (e.g., agent C_3) will increase their sociability, and thus their in-degree.

Observation 11 When agents prefer sociables and exchange many referrals, only the consumers who are initially close to providers get high in-degree. The remaining agents either have no in-degree or low in-degree.

On the other hand, if the referrals are selective (e.g., *Refer all matching*), agent C_1

will not necessarily locate agent C_3 in the beginning of the simulation. Hence, it will not replace agent C_2 with agent C_3 right away. Meanwhile, if agent C_2 learns about a provider that is beneficial for agent C_1, then agent C_1 will increase agent C_2's sociability more than anyone else's sociability. Even if agent C_1 learns about agent C_3 later on, it will not prefer agent C_3 over agent C_2. In other words, when the referrals are selective, there will be a few agents (e.g., agent C_3) that will have high in-degree because they start sociable. But, others (e.g, agent C_2) will also be likely to accumulate some in-degree after they discover useful providers. Figure 4.6 shows the distribution when the referrals are selective. Selective referrals allow local sociables to emerge over time. Hence, with selective referring, the consumers that do not start with an advantage of knowing providers can still end up with other consumers pointing at them.

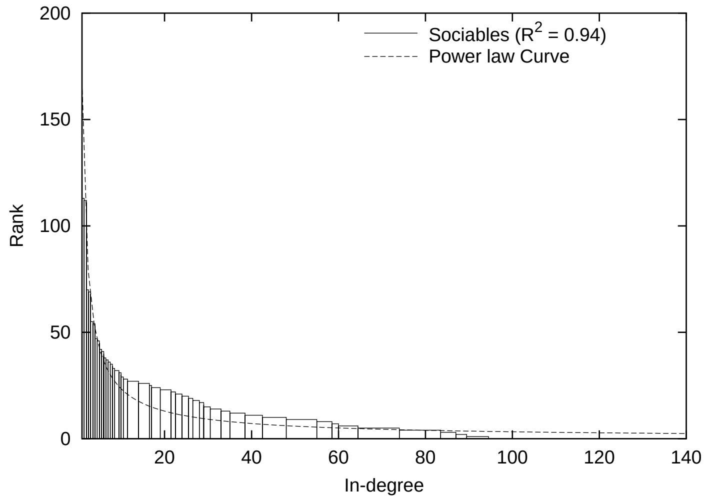

Figure 4.6: Power law distribution for *Sociables* policy with selective referrals

Observation 12 When agents prefer sociables but do not exchange many referrals, only a few consumers stand out as sociables. These agents get a substantially higher in-degree than the rest of the agents. The in-degree distribution follows a power-law.

4.2 Distribution of PageRanks

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 becoming authorities. We measure the authoritativeness of each agent using the PageRank metric (Equation 3.1) We study four factors that influence the PageRank distributions: the percentage of actual experts in the network, adaptability of the agents, and the referral and neighbor selection policies.

4.2.1 Percentage of Experts

Intuitively, the percentage of agents with high expertise plays a crucial role in the distribution of PageRanks. For example, when there are too many experts in the system, we expect that the PageRanks will tend to be shared among them. Having a small number of experts may ensure that experts with high authoritativeness will emerge. To study this point, we vary the percentage of the experts in the system. We study three populations with 5%, 10%, and 20% experts in them.

Figure 4.7 shows a histogram of PageRank distribution for three populations for PageRank values 2.5 and higher. The solid lines denote the population with 5% experts, the dashed lines denote the population with 10% percent experts, and the dotted lines denote the population with 20% experts. When the percentage of experts is high, the PageRanks are clustered for small PageRank values. For example, when the population has 20% experts, the number of agents having PageRank greater than 2.5 is higher than for the other two populations. For the greater values of the PageRank, the converse holds. For example, the only population that allows PageRank greater than 25 is the 5% expert population.

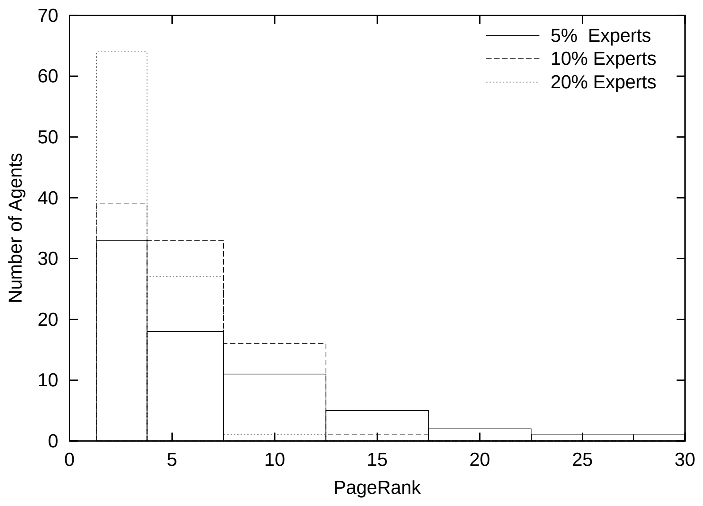

Figure 4.7: PageRank distributions for varying percentage of experts

Observation 13 There is an implicit competition among the experts. When there are too many experts, they end up sharing the incoming edges. Therefore, only a few receive a relatively high PageRank. When there are a few experts, those experts tend to dominate more clearly.

Since the population with 5% percent experts provide a broader distribution of PageRanks, we use this population for the following experiments.

4.2.2 Referral Policies

Next we study the effect of referral policies in the emergence of authorities. After each simulation run, the agents are ranked based on their PageRank. Figure 4.8 shows the PageRank distribution of the top 50 agents (out of a total of 400). If the agents use the *Refer all* policy, few authorities with high PageRanks emerge. For example, the 10th agent in the *Refer all* policy gets a PageRank greater than the first agent in two instances of the *Refer all matching* policy (with thresholds 0.4 and 0.5). Further, the *Refer all* policy creates a

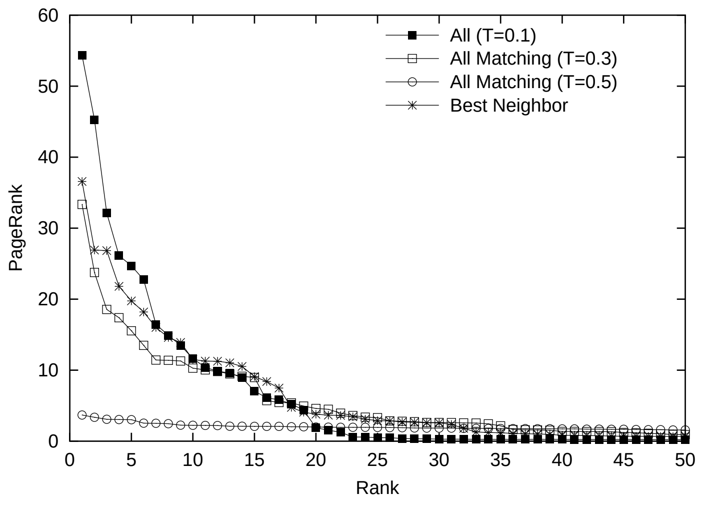

Figure 4.8: PageRank distributions for referral policies

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

Observation 14 Whereas more authorities emerge through *Refer all matching* policies, *Refer all* policy causes emergence of authorities whose level of authoritativeness is higher.

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

4.2.3 Adaptability

Throughout the simulation, after every few queries, each agent has an opportunity to choose new neighbors from among its acquaintances. Since agents learn about other agents through the answers they receive, changing neighbors allow them to point at agents that are expected to be more useful to them. This enables us to study the evolution of PageRanks, since PageRanks of the agents change as a result of the neighbor changes.

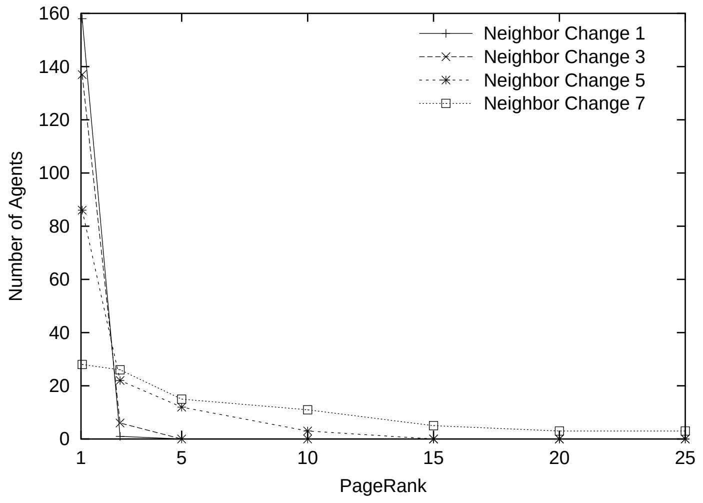

Figure 4.9: Change in PageRank distribution as agents adapt

Figure 4.9 plots the distribution of PageRanks after several neighbor changes. The plots correspond to the number of agents that achieve a PageRank higher than the value given on the X axis. To reduce clutter, we have omitted PageRanks smaller than 1. After the first neighbor change, most agents have a PageRank between 1 and 3. As the agents change their neighbors and point to the ones that are more useful to them, some authorities emerge. This is reflected in the graph by the increased number of agents that get a higher PageRank. After the final neighbor change, there are agents with PageRanks greater than 25.

Observation 15 As agents change neighbors, they do learn about the authorities and show a preference for linking to them.

4.2.4 Neighbor Selection Policies

Figure 4.10 plots the distribution of PageRanks with respect to neighbor selection policies. Again, the X axis shows PageRanks and the Y axis denotes the number of agents that achieve a PageRank greater than the PageRank shown on the X axis. The five plots correspond to *Providers*, *Sociables*, and three *Weighted average* neighbor selection policies with different weights.

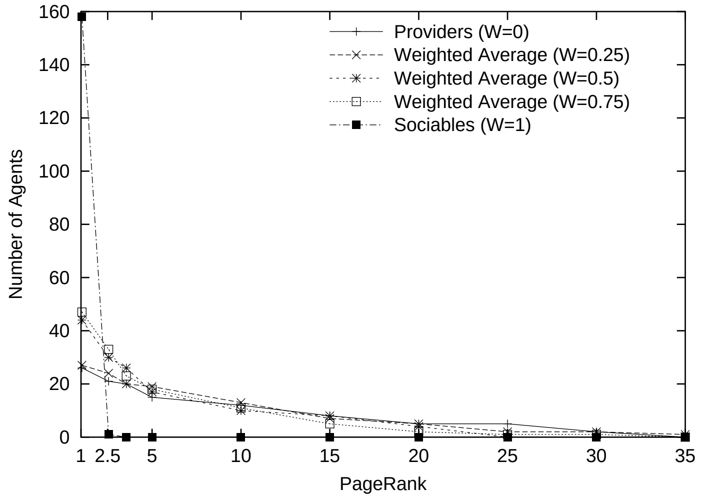

Figure 4.10: PageRank distributions for neighbor selection policies

All curves, except the one for *Sociables* policy, are similar to each other. In all four cases, the number of authorities that emerge is few. But the level of their authoritativeness is high. For example, for the *Providers* policy, 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 while slightly decreasing the number of agents with high authority. For example, with *Weighted Average* policy, when sociability and 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.

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

Observation 16 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, at the outset, the agents prefer sociables, there is little grounding in quality. Consequently, it is difficult to find good providers and thus strong authorities do not emerge.

Although only sociability does not help any agent to become authoritative in the beginning, once the network has stabilized, sociability helps as there is a basis for good referrals to be given and there is value in those who can give good referrals.

4.3 PageRank Distributions Under Perturbation

The factors mentioned in the previous section influence the PageRank distributions. As expected, the agents that get the most PageRanks are agents with high expertise or high sociability. Now, we manually modify the models of a few of the agents in the system to measure the consequences of the perturbation (vis à vis the existing PageRanks, which characterize the entrenched players) on the resulting population.

For example, what happens when an agent with a high expertise begins providing poor quality of service? Conversely, if an agent gains expertise over time, can it acquire high PageRanks and get a high ranking?

First, we run the simulation until the referral network stabilizes, i.e., there are no more neighbor changes. Then, we pick four agents that have ranked in the top 25 and switch their expertise values with four agents with low expertise. This will ensure that the four previously expert agents will start giving low quality service. Rather than choosing the top four agents, we choose four agents from the top 25 (ranked 1st, 4th, 16th, and 24th). This will allow us to see the impact of the initial position in the rankings.

We are interested in how the PageRank distribution and the ranking of these eight agents will change. Intuitively, we would expect the PageRanks to drop for the previously expert agents and to increase for the agents who have now become experts. To give a more complete analysis, we look at two orthogonal setups. In the first setup, agents use the *Sociables* neighbor selection policy, whereas in the second setup the agents use the *Providers* neighbor selection policy.

4.3.1 Agents Prefer Sociables

We first study the four agents that have started giving low quality of service (after their expertise is decreased). Figure 4.11 plots the evolution of PageRanks for two previous experts, agent 1 and agent 24. Interestingly, when agents prefer to be neighbors with sociable agents, the PageRanks of the two agents (4th and 24th) are affected more than those of the other two agents. More precisely, the absolute PageRanks as well as the rankings of these two agents drop.

The agent that has ranked 4th is now ranked at 18 and the agent that was ranked 24th is now ranked at 48. However, the other two agents (1 and 16) are not affected by the fact that they have started offering lower quality of service. Both their absolute PageRank and ranking stay almost the same with minor shifts between neighbor changes. This is unexpected. Looking closely at others' models of these agents, we see that even though agents

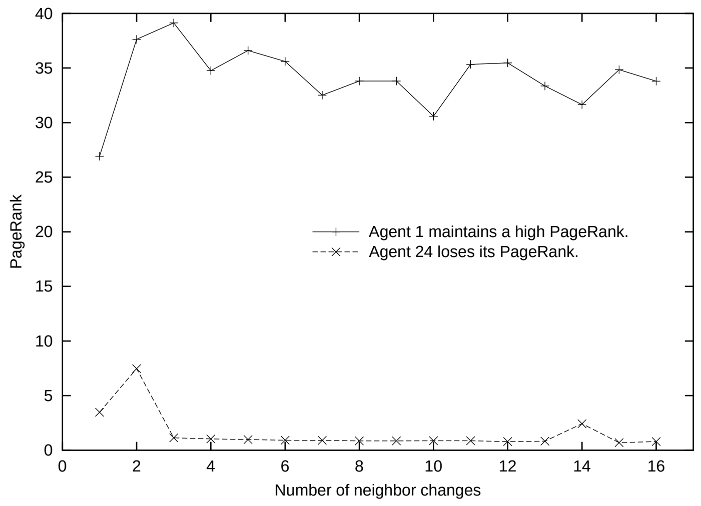

Figure 4.11: Evolution of PageRanks for two previous experts

1 and 16 have low expertise, they are still being kept as neighbors for their sociability. In other words, now that agents 1 and 16 cannot provide adequate answers, they start giving good referrals. This is reason enough for other agents pointing at these to keep them as neighbors. In other words, agents 1 and 16 keep their PageRanks because of their neighbors. On the other hand, since agents 4 and 24 do not have equally useful neighbors, there is no reason for other agents to point at them. What we see here is that when agents gain enough PageRank, they can maintain their ranking even after they lose their expertise as long as they maintain useful neighbors.

Meanwhile, the four agents who have now started offering good services can increase their PageRanks only slightly; the greatest increase in PageRank was of 1. Since the other agents weight sociability more than expertise, these four agents are not being chosen as neighbors as much as might have been expected, even though these agents have high expertise now.

Observation 17 Even though an agent loses its expertise, it may still be viewed as an authority because of its sociability.

4.3.2 Agents Prefer Experts

The previous section explained how the PageRank distribution can evolve when the agents prefer sociable neighbors. In this section, we look at the converse case: how does the distribution evolve if the agents choose experts over sociables?

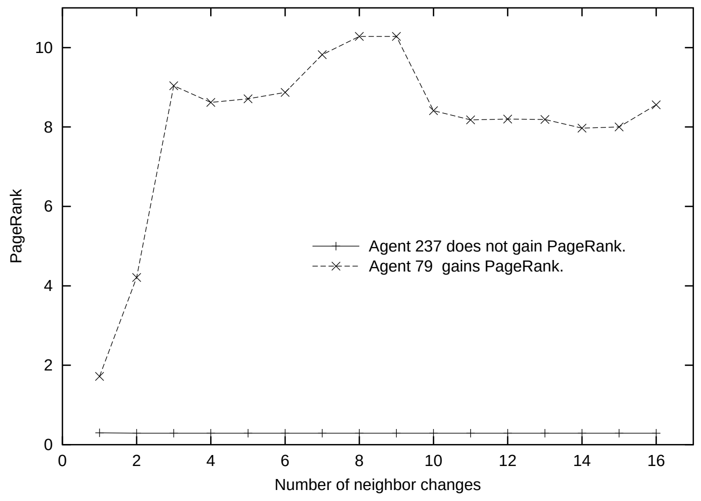

Figure 4.12: Evolution of PageRanks for two new experts

Again, we look at the four agents who have stopped providing high quality of service. This time, all of their PageRanks, and hence their ranking, drop slowly. The interesting cases occur for the other four agents who have now started providing high quality of service. Figure 4.12 plots the evolution of PageRanks for two of the agents.

Agent 237 cannot improve its PageRank at all. The reason for this is that only a few other agents point at it. Therefore, it is not referred to other agents. Over all, even though this agent has higher expertise than most of the agents that rank above it, it cannot get enough PageRank to improve its relative ranking.

This is not the case for agent 79. It gets discovered by other agents and hence can substantially increase its PageRank. However, there is a limit to this increase. That is, even though it accumulates a high PageRank, it cannot make it to the top ten. This is caused by the stability of the other agents' neighbors. Most other agents have already found good service providers for their needs, so they rarely change neighbors.

Observation 18 Even though an agent gains expertise, it may never be viewed as an authority since its initial position in the network may be poor, i.e., not well-connected.

4.3.3 Does the Winner Take All?

As noted above, the distributions of in-degrees, out-degrees, and PageRanks all follow power law distributions. These distributions suggest some sort of preferential attachment wherein Web links are likelier to be made to those pages that already have a high in-degree or PageRank. Hence, the mottos "the rich get richer" and the "winner takes all" are used to describe this phenomenon. However, Pennock *et al.* [2002] recently discovered that the reality is closer to a combination of preferential and uniform attachment, meaning that the winner doesn't always "take all".

Our referrals-based approach allows us to model the process through which the links among agents emerge. Thus, we can explicitly model the various policies through which the referrals are generated and through which the agents update their sets of neighbors. This enables us to capture various shades of trust formally. Specifically, from the above experiments, we conclude that authorities with high PageRanks emerge when (1) there are few experts in the system, (2) agents exchange more referrals, and (3) agents change neighbors based on other agents' expertise and sociability. The PageRank distribution then evolves based on how these authoritative agents maintain their quality of service. This evolution is highly influenced by whether agents prefer sociables or experts. When agents prefer sociables, agents who have gained high PageRanks can sometimes keep their PageRank even without providing high quality of service. On the other hand, when experts are chosen over sociables, the agents with high PageRanks must continue to offer high quality of service in order to maintain their PageRanks.

Similarly, when sociables are preferred, there is little chance for newcomer experts to get high PageRanks since they are not pointed to by any sociable agents. Preference for experts relaxes this case but still does not guarantee high PageRanks for newcomers. These findings lead us to conjecture more broadly that in environments where sociability dominates (with respect to expertise), the winner will take all, whereas in settings where expertise dominates, the winner may not take all. If new services emerge and draw away some of the demand then there will be greater churn in the rankings of the top-most authorities.

Chapter 5

Emergence of Trust

Previous agent approaches for trust emphasize either its local or its social aspects [Barber and Kim, 2001; Sen and Sajja, 2002]. The referrals-based approach provides a treatment for both local and social trust. Local trust is established as a result of evaluation of services and referrals. That is, an agent trusts another agent that provides a useful answer or a useful referral for a particular query. Social trust is accomplished through referrals. By following referrals from agents that it trusts, each agent can locate other trustworthy parties for its needs.

Here, we propose a graph-based representation for services that enhances the treatment of local trust. The representation helps agents to reason about their interactions with others locally and in a more systematic way than the existing vector approach. This representation enables us to address two properties of trust that are not adequately addressed by current approaches. One, trust often builds up over interactions. That is, you might trust a stranger for a low-value transaction, but would only trust a known party for a high-value transaction. Two, trust often flows across service types. That is, you might assume that a party who is trustworthy in one kind of dealings will also be trustworthy in related kinds of dealings.

The rest of this chapter is organized as follows. Section 5.1 introduces a graph-based representation for agents to model others. Section 5.2 describes our experimental setup. Section 5.3 compares this graph-based representation with a vector representation in terms of efficiency and effectiveness. Section 5.4 discusses and experimentally evaluates factors that are related to performance.

5.1 Graph-Based Service Representation

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

In the previous chapters, the set of services was represented through a vector space model, where each element in the vector corresponds to a different domain and the weight of the element denotes the fitness of the service for that domain. The vector representation is simple and quite effective if the elements are independent, since a vector representation does not capture any relationships between vector elements.

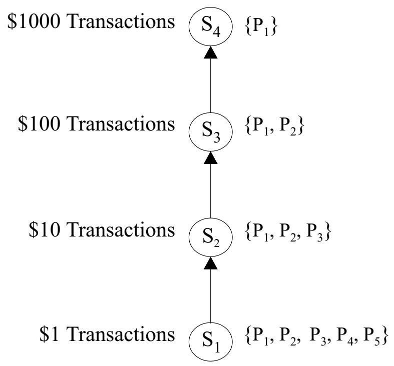

Figure 5.1: A totally-ordered service graph

Figure 5.1 shows a simple graph. Here, each node represents transactions of different values. S_1 denotes transactions worth $\$1$, S_2 denotes transactions worth $\$10$, and so on. The list next to each node represents the trustworthy providers for that node. The agents trusted for a node are a subset of the agents trusted for the lower node. That is, if you trust someone for a $\$10$ transaction, you trust him for a $\$1$ transaction as well (e.g., P_3). The reverse need not hold. You might trust many for transactions of $\$1$ but probably only a few for $\$1000$ transactions (e.g., P_1).

Here, we propose to represent the services using a graph. The graph representation is more expressive than a vector representation because it can capture relationships between service types that a vector representation cannot. For example, a service provider that has been found to be trustworthy for one type of service can be considered for another type of service based on how well the services relate.

A *service graph* is simply a graph whose nodes map to service types. Any two services that are related are joined by an edge. Here an edge $\langle s_i, s_j \rangle$ indicates that a provider who can perform s_i well may also be able to perform s_j well. The extent of the relation between the services is given through the weights on the edges. The weights can range between 0 and 1. The bigger the weight, the more relates the services are. Conceptually, a graph represents how the services relate to each other in terms of difficulty. If there exists a directed path between two service types s_i and s_j, then a provider of s_j can also perform s_i to some degree. Figure 5.2 illustrates an example service graph, with nine service types. Any service provider who can perform S_8 can also perform all other services in the service graph.

When an agent needs a provider for a service for which it knows of no providers, it can potentially ask others as before or *promote* a provider that it has used for another service. Promotions provide a systematic way to reuse previous experiences with the service providers. A provider is tried for a new service only if it has performed well for another

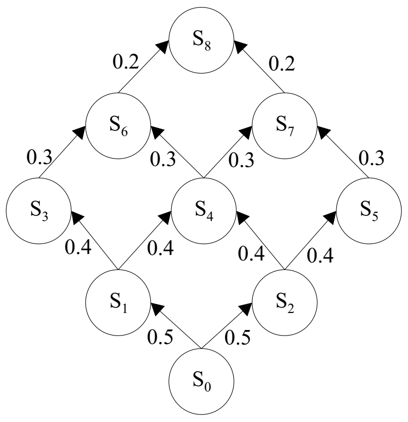

Figure 5.2: An example service graph

service, and if performing well in the first service indicates that the provider may perform well for the second service. The likelihood of a service provider in a lower node to perform a service in the upper node is represented by weights on the edges. For example, the weight 0.5 from S_0 to S_1 means that a provider of S_0 will likely be providing S_1 half the time.

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

Algorithm 3 outlines which service providers an agent promotes to service j. When promoting a provider from s_i to s_j, two factors are considered: the trustworthiness of the provider for s_i, and how related s_i and s_j are. We calculate the trustworthiness of the provider p at s_i (t_{pi}) through its ratings at s_i and the number of interactions for s_i. The strength of the relation between s_i and s_j is given by the edge weight, w_{ij}.

$$(w_{ij} \times t_{pi}) > \theta \tag{5.1}$$

The product of the edge weight with the average ratings projects how much the agent can reproduce its ratings in s_j, as given in Equation 5.1. If this projected value is greater than a promotion threshold θ, then the agent can be promoted to perform s_j (line 4).

Algorithm 3 promoteLocally(j)

```
1: for $i = 1$ to $|nodes|$ do
2:    for $k = 1$ to $|providers(i)|$ do
3:       $p$ = providers($i$)($k$)
4:       if ($t_{pi} \times w_{ij} > \theta$) then
5:          if (numberOfInteractions($p$)$\geq \lambda$) then
6:             Add $p$ to promotedProviders
7:          end if
8:       end if
9:    end for
10: end for
11: return promotedProviders
```

In the extreme case, if $w_{ij} = 0$ (the services are not correlated), then the service provider is not expected to perform well in s_j even if it performs well in s_i. Conversely, if a provider is not trusted for s_i ($t_{pi} = 0$), then the provider will never be promoted to s_j irrespective of how correlated the two services are. A third factor considered is the number of previous interactions with the provider at s_i (line 5). An agent may require to promote providers only after a certain number of interactions has occurred.

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

5.2 Experimental Setup

The experiments use 100 service consumers. The number of service providers is 32 and the number of service types is 9. Initially, each agent knows two randomly chosen service providers. We experiment with different starting service graphs. After a service graph is generated, the edges on the graphs never change, but the weights can be updated by each agent as seen fit. Each agent is neighbors with three randomly picked agents. Each query denotes the desired service type, e.g., S_0, S_1, and so on. Note that not all service providers offer all the services. The key property we want to capture in modeling the distribution of the service providers for different services is that in real life, we would expect more service providers to offer the easier services than the harder ones. Hence, the number of providers would decrease as the service gets more specialized. With this intuition, the experiments are set up such that most of the service providers can perform services that are lower down in the service graph, whereas only a few of them can perform harder services, e.g., S_8, the most specialized service in Figure 5.2. We capture this intuition by decreasing the number of providers approximately by half between two consecutive nodes. For example, when the number of service providers is set to 32 and number of service types set to nine, while 15 service providers offer service S_1 only 7 of them provide S_3. The number of service providers for each type of service for this graph is given in Figure 5.3.

The second modeling characteristic we capture is the distribution of the requests for

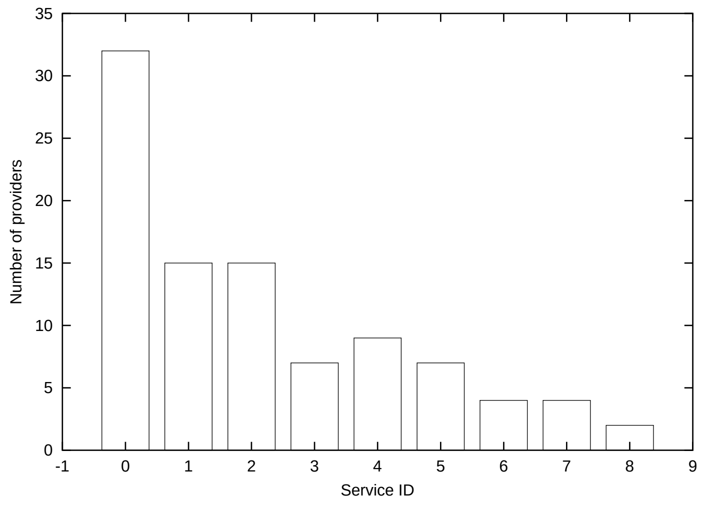

Figure 5.3: Distributions of the service providers

different services, that is, the differences in the frequency of requests for a particular service from the service consumers. Again in real life, we would expect most requests to be for services with intermediate difficulty rather than for very easy or very hard services. For this reason, we use a normal distribution to model the frequency of the incoming requests. For example, for the service graph given in Figure 5.2, this distribution implies that services S_0 and S_8 get the least number of requests, whereas services S_3, S_4, and S_5 get the most requests.

For these experiments, the service providers provide services consistently. That is, a service provider that has provided a service will again perform the same service. For each experiment, we report averages from three simulation runs (additional runs yield similar results).

5.3 Comparison of Representations

We compare the service graph representation with the vector representation in terms of effectiveness and efficiency. A representation is effective if it allows agents to find the desired service providers. A representation is efficient if it allows the service providers to be found with as few messages as possible. In order to compare the effectiveness and the efficiency of the two approaches, the simulations are run with an identical setup. Each agent generates 30 queries and may change neighbors after every 3 queries. The agents with service graphs use the graph in Figure 5.2 as the initial service graph.

Effectiveness is measured by the percentage of the queries that have resulted in finding a useful service provider. That is, the ratio of queries that lead to useful service providers to all the generated queries is calculated. We study the effectiveness after every five queries for the service graph approach and the vector approach. To give a more detailed analysis, two cases of the vector approach, one with referrals, one without referrals, are considered.

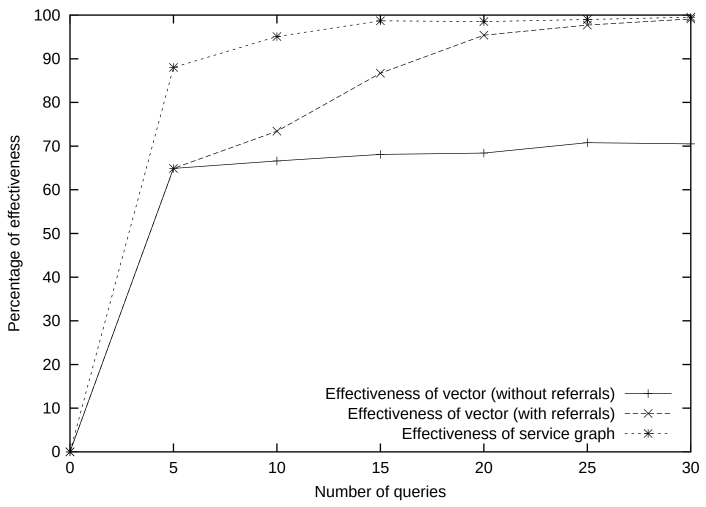

Figure 5.4: Effectiveness of the representations

Figure 5.4 plots the average effectiveness of all 100 agents after every five queries.

During the first five queries, the agents that use a vector with referrals perform no better than the agents without the referrals. Agents that employ a service graph (dotted line) achieve higher effectiveness percentage than both of the vector approaches. Between the fifth and the twentieth queries, the effectiveness of the agents that use a vector with referrals increases considerably. Even though the agents with the service graphs still achieve higher effectiveness rates, now the difference between the vector (with referrals) and the service graph is quite small.

The performance of the vector approach increases as the agents learn about their neighbors and change their neighbors accordingly. After the 30th query, both approaches achieve an effectiveness rate of 99%. However, when referrals are not employed, the effectiveness of the agents barely increases (solid lines). The average effectiveness for the no-referral case oscillates between 63% and 70%. Having no referrals presents two disadvantages to the agents. One, obviously they can pose their queries only to their neighbors, and incompetent neighbors cannot provide answers. Two, since there are no referrals, the agents interact with few other agents and learn about only a small part of the society. Hence, when they change their neighbors, the set of agents they choose from is small and pseudo-random.

Next, we compare the average number of agents contacted per query (over 30 queries). Figure 5.5 plots this efficiency value for both approaches. The agents that use a service graph contact fewer than one agent on average. Moreover, this number consistently drops as they generate more queries. The agents that use a vector approach with referrals contact slightly more than three agents on average even after they generate many queries. The agents that use a vector without referrals contact fewer than three agents on average but, as for the graph approach, the number of contacted agents drops as the number of queries increase. Overall, for the vector representation the addition of referrals increases the number of contacted agents for the benefit of increased effectiveness, whereas the service graph

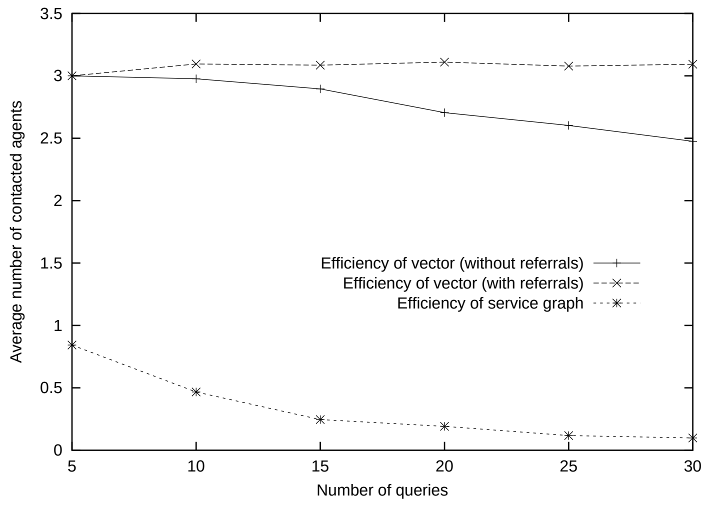

Figure 5.5: Efficiency of the representations

outperforms the vector representation in both effectiveness and efficiency.

Observation 19 Consumers can locate trustworthy service providers more effectively and more efficiently with a service graph representation than with a vector representation.

5.4 Evaluation

We study how the initial setting, promotion threshold, and the number of previous interactions affect promotion accuracy and effectiveness in finding trustworthy service providers.

5.4.1 Control Variables

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

The second factor is related to how much the agents are initially willing to try new service providers. This factor, termed *trust prejudice*, captures whether an agent is willing to trust newcomers [Jonker and Treur, 1999]. We capture this intuition through the initial graph weights. For example, if initially all the weights are 1, then the agents are willing to try out all new service providers in all types of services. Conversely, when the weights are all 0, the agents have the prejudice that none of the agents can be trusted.

Service graphs are evaluated using three initial settings. In the *homogeneous* setting, each agent starts with the service graph shown in Figure 5.2. In the *trusting* setting, the graph edges are the same but the weights are higher (meaning the agents trust others more). In the *heterogeneous* setting, each agent starts with random weights on random edges of its choice.

Promotion Threshold The estimated weight between two services is adjusted based on previous promotions between the two services. Intuitively, the promotion threshold denotes how much risk an agent is willing to take in its promotions. If the threshold for promoting is low, then the agents will promote more providers, but might find out that more of these providers cannot perform the service. On the other hand, if the agents are reluctant to promote, then they might miss a chance to find a provider for a desired service. (In Algorithm 3, θ refers to the promotion threshold.)

Number of interactions The overall rating of a provider for a service should be reliable. It is widely accepted that the number of previous interactions increases the accuracy of the trust assessment [Falcone and Castelfranchi, 2001]. That is, the average rating may not be representative if the total number of interactions are few. In other words, a service provider with a ranking of 0.7 over three interactions might be trusted more than a provider with a ranking of 0.8 over one interaction. More formally, previous interactions with a provider allows an agent to derive a confidence interval for the provider's performance for the service. Statistically, when fewer data points are considered, the range of the confidence interval will be larger than that of the case with more data points [Hinkle et al., 1979]. This is the intuition behind building trust over interactions.

In our approach, agents use the number of interactions as a gating factor. Only those providers that have proved sufficiently trustworthy in another service, which is sufficiently closely related to the service under consideration, and such that the agent has interacted with these providers often enough to trust them adequately, are considered for promotions. (In Algorithm 3, λ refers to the required number of interactions.)

5.4.2 Results

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

Figure 5.6 plots the promotion error for varying promotion thresholds. For all three curves, the error drops when the promotion threshold increases. That is, when agents take fewer risks, they make fewer mistakes. The heterogeneous setting has higher weights for more edges than the other two setups, and hence allows more promotions. For this reason, it is more prone to errors.

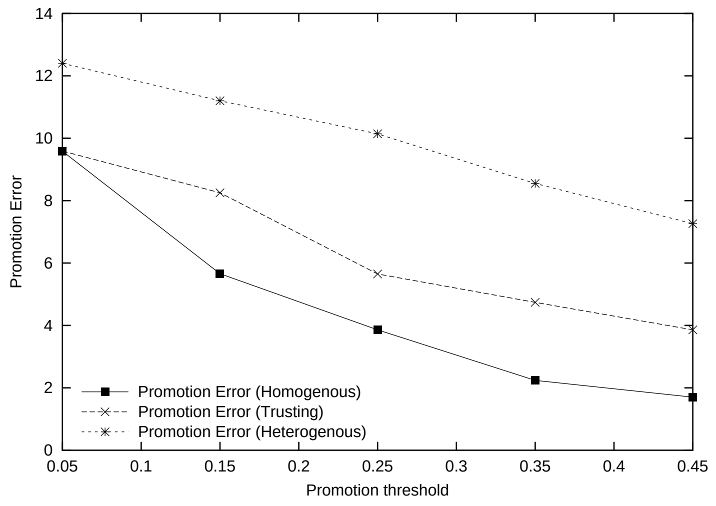

Figure 5.6: Effect of initial setting on promotion accuracy

Observation 20 When promotions are made selectively, agent make fewer promotion errors.

Next, we study the effect of number of interactions on promotion error. For each value of the promotion threshold, we plot the average promotion error. Figure 5.7 shows three plots for the homogeneous setting, corresponding to one, two, and three required interactions prior to promotion. The promotion error decreases with the number of previous interactions. For a threshold of 0.25, for example, when the required number of previous interactions is just one, the promotion error is almost 6. When the number of interactions is increased to two, the error drops below 4. When the number of interactions is further increased to three, the error becomes less than 2. In all three curves, increasing the promotion threshold decreases the promotion error, though the improvement is more significant for fewer interactions.

Observation 21 Increasing the number of previous interactions decreases the promotion error.

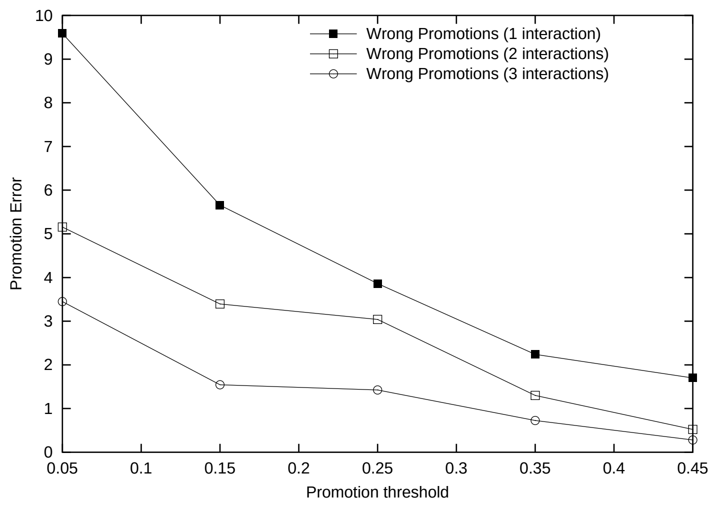

Figure 5.7: Effect of previous interactions on promotion accuracy

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

Again, we first look at the effect of initial setting on the effectiveness. Figure 5.8 plots three effectiveness curves for the three initial settings. This time the random setup achieves higher effectiveness than the other two setups. Since the random setup assigns weights to many edges, and hence allows more promotions, many providers—useful or not—are promoted and tested, resulting in a provider almost always being found.

Figure 5.9 plots three effectiveness curves for varying values of the promotion threshold using the homogeneous initial setting. Again, each curve corresponds to a case where a different number of previous interactions is required. Independent of the number of interactions, if the threshold is high, the effectiveness is very low. Interestingly, for smaller values of the threshold, we see that agents achieve a higher level of effectiveness (find more trustworthy agents to interact with) if the number of interactions is lower. This is the

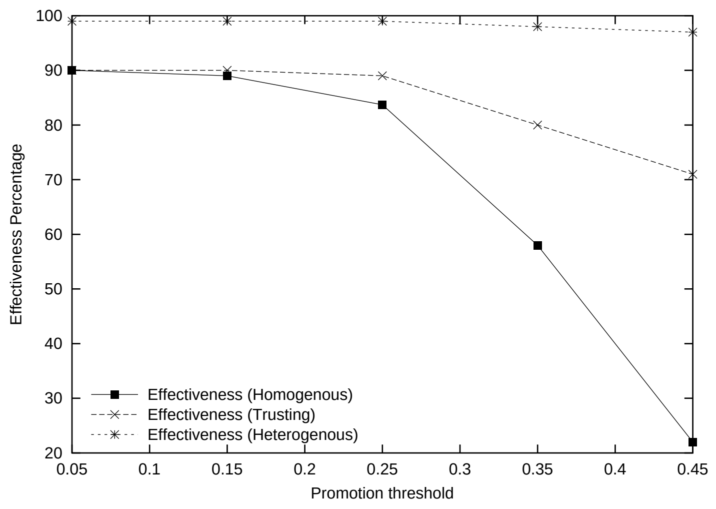

Figure 5.8: Effect of initial setting on effectiveness

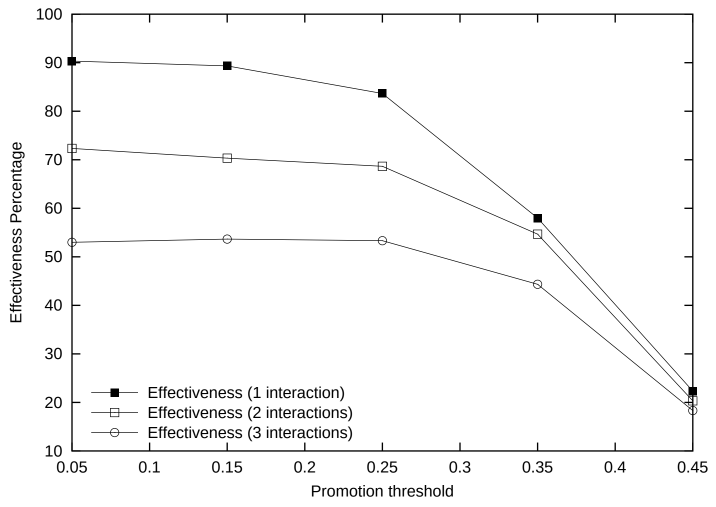

Figure 5.9: Effect of the previous interactions on effectiveness

opposite of the curves for the promotion accuracy, where we saw that a greater number of interactions decreases the promotion error. In other words, high promotion accuracy rarely coexists with high effectiveness. For example, in Figure 5.7 when the number of previous interactions is set to three (with threshold 0.35), the promotion error is below 1. However, effectiveness for the same setup is not even 50%.

Observation 22 There is a trade-off between finding more trustworthy agents and promoting only the trustworthy agents.

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

Performance $=$ Effectiveness $\times$ Accuracy

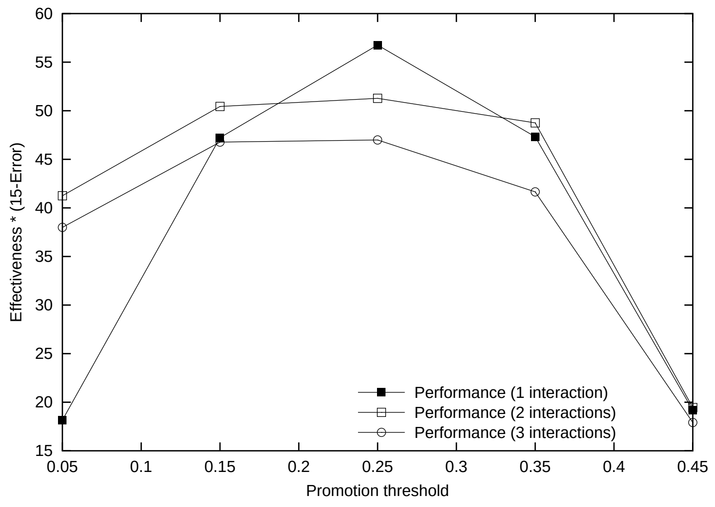

Figure 5.10: Effectiveness and promotion error trade-off for homogeneous setup

Figure 5.10 plots this performance value based on Figure 5.7 and Figure 5.9. Neither extremes of the promotion threshold (0.05 and 0.45) achieve high performance. The lower threshold suffers from high promotion error, while the high thresholds lacks effectiveness.

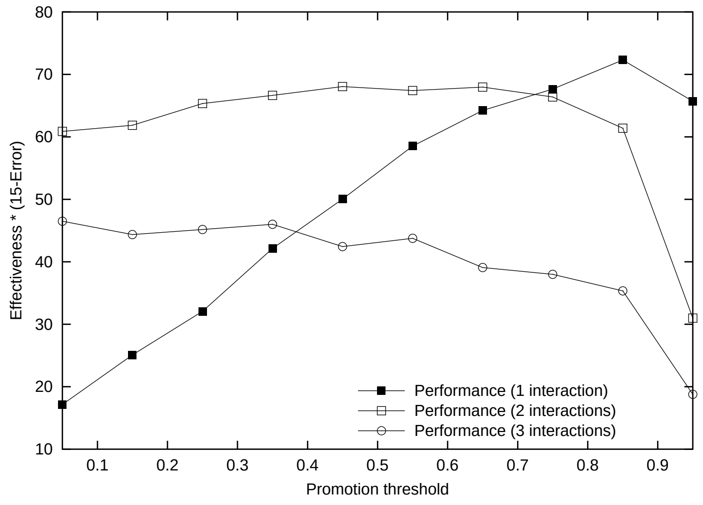

Figure 5.11: Effectiveness and promotion error trade-off for heterogeneous setup

Optimal performance lies in the middle values of the promotion threshold. Among these, the performance is always better when the number of interactions is either 1 or 2. This suggests that the third interaction does not add much value to the performance. Among the 1 and 2 interaction cases, except for one value of the threshold (0.25), the 2 interaction case outperforms the 1 interaction case. Similarly, Figure 5.11 plots the performance for the heterogeneous setup. Again, the performance is lower on the boundary thresholds and the optimal performance lies in the middle. Because the initial weights of the heterogeneous setup are high, the optimal performance is shifted towards higher thresholds. In general, these results suggest that it is better to be less cautious, trust more, and make some mistakes to be able to exploit a wider range of promotions.

Chapter 6

Discussion

Our approach takes an adaptive, agent-based stance on peer-to-peer computing. Our philosophical claim is that referrals are essential for locating services. Referrals capture the manner in which people normally help each other find trustworthy authorities [Nardi et al., 2000]. This is an important motivation for referral systems. Whereas the importance of referrals have been known in the real world, referral systems have come around recently. Next, we review some of these systems as well as other work related to our approach.

6.1 Related Work

Multiple Intelligent Node Document Servers (MINDS) was the earliest agent-based referral system [Bonnell et al., 1984]. 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.* [1997] 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. Our work, on the other hand, tries to uncover the structural properties of the network to design mechanisms that will improve the quality of the network.

Yu and Singh [2003] study referral networks in the context of scientific collaborations. They study different aspects of referral networks, in particular 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 the most promising referrals, i.e., referrals with high weights.

Next, we discuss the literature on P2P systems, Web structure, and trust with respect to our approach.

6.1.1 P2P Systems

Plaxton *et al.* [1997] design a randomized algorithm to search for possibly replicated objects in a distributed environment. The algorithm tries to minimize the cost of an object access by essentially minimizing the distance between the peer that is looking for the object and the peer that hosts the object. Each peer keeps a set of shared objects. The access scheme allows read, insert and delete operations on these objects.

In addition to the entries, each peer maintains a neighbor table and a pointer list. The neighbor table is divided into levels that hold primary, secondary, and reverse neighbors. The pointer list holds pointers to the copies of the objects on the network. When searching for a peer that has a copy of the desired object, the peers on the path from the origin to the destination exchange information about the upper bounds on the costs of likely candidates. Thus, each peer constantly updates its list of pointers to reflect the newly discovered replicas.

When a peer inserts a replica of an object, it contacts its reverse neighbors to inform them of the replica and the replica's cost. If this replica is cheaper than what the neighbors had before, then the neighbors update their pointer tables and propagate the information on the neighbor chain. Similarly, when a peer deletes an object, it contacts its neighbors to signal the deletion. The peers that have this object are obliged to delete it. That is, the autonomy of the peers is restricted.

The intuition of Plaxton *et al.*'s algorithm has been reformulated in recent peer-to-peer network architectures [Stoica et al., 2001; Ratnasamy et al., 2001; Rowstron and Druschel, 2001a; Aberer, 2001]. These systems all 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 one unique peer that is responsible for holding it. First, we we review these systems more in detail, and then compare them to our approach.

Stoica *et al.* [2001] develop Chord, a distributed look-up protocol that maps keys to peers in a P2P system. Chord uses a variant of consistent hashing to balance the load on the peers, i.e., to ensure that each peer gets approximately the same number of entries. Each peer is assigned an identifier by hashing its IP address and the peers are ordered in an identifier circle. If the identifier of peer n equals the hash value of key k, or if n follows the peer whose identifier equals the hash value of k, then n is called successor of k and becomes responsible for key k.

Each peer keeps track of the peers that comes before and after itself. In addition, each peer has a finger table with at most m entries, where m is the number of bits in each key. The ith entry (ith finger) in the finger table shows the peers that are the successor of keys that are 2^{i-1} bigger than the peer identifier. When a request for a key comes in, a peer checks its finger table to find the peer that is responsible for the key. If it cannot find such a peer, it forwards the request to the peer whose identifier is closest to the key. Assuming that the circle never gets disconnected, this search technique ensures that the peer that owns the requested key is always found. When a new peer enters the system, it is assumed that it knows at least one other peer. Through that peer, it learns its predecessor peer as well as its finger table. Some keys that used to be mapped to other peers are now owned by this new peer. Thus, other peers in the system must now update their finger tables to account for this new peer.

Ratnasamy *et al.* [2001] design an indexing scheme that maps keys to the peers that carry the file name associated with the key. This indexing scheme is termed a content-addressable network (CAN). A CAN allows insert, look-up, and delete operations. The whole network is envisioned as a big hash table where each peer is responsible for maintaining a small zone. The entry space is a d-dimensional Cartesian coordinate space on a d-torus. A $\langle$key, file$\rangle$ entry maps onto a point P in this coordinate space. The peer that owns the zone to which P belongs is responsible for this entry. In addition to the entries in its zone, each peer maintains a list of neighbor zones and their addresses. When looking up a key, a peer first checks to see if the key falls into its zone. If that's the case, the file that maps to this key is returned. Otherwise, the peers sends the request to the neighbor whose coordinates are the closest to the key.

The system is self-organizing, in that zones are assigned to peers at run time. When a new peer enters the system, one of the existing zones is divided into two so that one half can be maintained by the old resident and the other half by the new peer. After such a split takes place, the neighbors of the zone are notified of the split so that each peer can update its neighbor tables. Similarly, when a peer becomes unavailable, one of the adjacent peers takes over the zone that was left behind. Possible extensions to this general framework are multiple coordinate spaces that increase information availability, overloading of zones by allowing a zone to be owned by multiple peers, accommodating multiple hash functions that enable an entry to be mapped onto several points in the coordinate space (to increase fault tolerance), and several metric optimizations such as joining a large zone rather than a purely random zone, or minimizing the distance between neighbors in the network.

Rowstron and Druschel [2001a] design Pastry, a layer that enables object location and routing in peer-to-peer systems and PAST [Rowstron and Druschel, 2001b], a system implemented on top of Pastry. Pastry is quite similar to CAN in its design. Again, systems that need to locate and route objects can use Pastry as an underlying layer. The objects are located by routing queries based on the prefixes of the keys, similar in spirit to Plaxton *et al.*'s algorithm. However, rather than minimizing a generic cost function, Pastry exploits local proximity by minimizing a scalar proximity metric.

Each peer in the Pastry system has a unique identifier called a node id. The peer that receives a message and a key forwards both of these information to the peer whose node id is closest to the message key (among all peers that are online at the time). Among possible peers whose ids are close to the message key, the Pastry system always chooses the peer with the smallest distance between the initial peer and the destination peers. Thus, at each step the message travels the minimum possible distance. Of course, this does not necessarily guarantee that the overall distance that the message travels is the shortest possible path.

Each Pastry peer maintains a routing table, a neighborhood set, and a leaf set. The routing table contains n rows, where the ith row contains the node ids of the peers that differ from the current row in the $(i + 1)$th bit. The neighborhood set contains the node ids and IP addresses of the m closest peers. The leaf set holds the node ids of the agents that are in a certain upper and lower proximity of the original peer's id. When a message needs to be routed, the peer first checks its leaf set to see if any node ids there match the message key. If that's the case, the message is forwarded to that peer. Otherwise, the peer searches its routing table to find the peer which is closest to the message key. If such a peer is not found, then the message is forwarded to a peer whose id is at least as close to the key as this peer, but whose node id is greater than the current peer's id.

The system accommodates arrivals and departures of peers. When a new peer arrives, it is assumed to know at least one other peer. There is a special join operation that enables the new peer to get the state tables of peers whose ids are closer to the node id of this new peer. A peer is said to have failed when its immediate neighbors cannot communicate with it anymore. The failed peers in the leaf sets have to be replaced with other peers. To find peers to make this replacement, a peers asks for the leaf set of the peer that has the greatest node id on the side of the failed peer. The leaf set is updated based on the leaf set of this peer. Both the routing table and the leaf set are used for locating peers, whereas the neighborhood set is used to exchange information about nearby peers. Each peer periodically pings its neighbors to see if they are still alive. If not, it requests the neighborhood sets of one of its live neighbors to find out about new peers and to update its own neighborhood set.

Fiat and Saia [2002] develop a censorship-resistant content-addressable network using the content-addressable network of Plaxton *et al.* The aim of this system is to enable access to most data items even after half of the nodes are removed from the network. The network is organized as a butterfly network with each node called a supernode. Every supernode is assigned a set of nodes. The supernodes at the bottom of the butterfly (bottommost supernodes) carry the data items. A query starts from a set of topmost supernodes and is performed in parallel. From a topmost supernode, the query follows the middle supernodes to get to one of the bottommost supernodes. If this bottommost supernode contains the data item, then the item is returned through the same path.

Aberer [2001] develops P-Grid, an access structure for P2P systems. Similar in spirit to the content-addressable networks, P-Grid is based on peers owning part of the key space. Initially, each peer is assumed to own the whole space. When two peers meet, they divide the space in two. Thus, as the peers organize, each peer owns only a part of the key space. Contrary to the content-based networks, when a peer goes offline, it does not hand over its key space to the other peers. Although this keeps the network traffic low, that part of the network is not reachable unless it has been replicated by another peer.

There are major differences between these distributed-hash table approaches and our referrals approach.

- These systems do not preserve the autonomy of the peers. Since the keys are assigned to peers rather than allowing the peers to choose the ones they want, the peers in these systems are not autonomous. It is assumed that whenever a peer receives a query it will act in the predefined manner.
- Each peer in these systems has a table that aids the search when the item being sought does not reside at this peer. This is similar to our neighbors concept. However, in our approach, each peers models those it interacts with in terms of expertise or sociability. Hence, in our approach, each agent can contact only those neighbors that are likely to answer a given query.
- These systems assumes that all peers are trustworthy which is a major assumption for open systems. The peers in these systems do not model others based on their quality of service. In our approach, each agent can build models of others and contact only those that are likely to answer a query based on prior interactions or referrals from trusted parties.
- These systems are not adaptive. In our approach every agent can change its

neighbors if it finds other agents that are better matches for itself. The aforementioned systems lack this adaptability. First, the routing is defined deterministically. Second, peers cannot change their neighbors, unless the neighbors go off-line. That is, even if a peer is not satisfied with the quality of the service it receives, it still needs to contact the same peer for a query.

The Cooperative File System (CFS) [Dabrek et al., 2001] is a system that uses Chord as a lower layer. Each file in CFS is divided into blocks and each block is stored on a different server. There are two reasons for this division. First, this ensures better load balance among servers. Second, there might be servers that have little storage, and thus cannot keep any large files. By dividing the files into blocks, even the servers with small capacities can be utilized.

The CFS system has two layers. The first layer is the Chord layer, which takes care of mapping from blocks to servers. The layer on top of Chord is called the DHash layer, which is responsible for distributing and replicating blocks on different servers. DHash automatically replicates each block on the k servers that follow the original server in the successor list. Hence, when the actual server is down or busy, the peers can go to one of the k successor servers and get the block. In addition to this, at the end of each search operation, the block that is sought is cached at all intermediate peers. That is, the block is cached by all the peers that have helped locate it. The idea is that these peers are likely to be contacted to find the same block. Hence, in addition to the k replications of each block, there are numerous cached copies in different peers. When a block is updated, it is easy to update the k replicas since they lie on the k successor servers, whereas it not clear how all the cached copies are located and updated.

OceanStore [Kubiatowicz et al., 2000; Rhea et al., 2001] is yet another global storage application that uses a peer-to-peer network. Two features of OceanStore that set it apart from other storage systems are partial handling of untrusted peers and exploitation of locality. Contrary to other storage systems, OceanStore has an access control that regulates the reading and writing of data items. In other words, only some of the peers can read and write certain items in the system. Unlike CFS, data items can reside on any server in the system. The search for an item is performed in two steps. First, a probabilistic algorithm is used to search the nearby peers. If the desired item is not located close by, then the second algorithm, which is a variant of Plaxton *et al.*'s algorithm, is used.

OceanStore allows modification of replicas by multiple parties, e.g., two peers can modify two different replicas. To cope with possible inconsistencies, a conflict resolution scheme is used. Each replica of a data item is considered as primary or secondary. The difference between these two is that the primary replicas exist on trusted peers and always have the correct version of the data item, whereas the secondary peers may have out-of-date versions of the item. All requests for updates to an item go to the primary tier of peers, which communicate through a Byzantine agreement protocol to decide on the order of updates. Meanwhile, the peers that request the updates start to send the updated version of the item to the secondary peers. After the peers of the primary tier finalize the data item, they distribute it to all the secondary peers.

Similar to the censorship-resistant network of Fiat and Saia, OceanStore has a mechanism to enable fault tolerance, named deep archival storage. It employs erasure codes, which divide a given entry into n fragments and then transform these n entries into a desired number of entries, say $4n$. When the item needs to be reconstructed, any n fragments are sufficient. This means that even if some of the peers that host the fragments are down, the item can be reconstructed. Here again, fault tolerance is achieved at the expense of violating peers' autonomy. Both these systems assume that any file fragment can be readily placed in any peer, regardless of the peers' choice.

6.1.2 Web Structure

Watts [1999] showed that a social network becomes a *small world* when the clustering is high but the characteristic path length is small. In Watts' work, the clustering happens as a result of a random function: some edges are relocated randomly after beginning with an initial ring structure. Here, we do not enforce any initial starting structure; we let each agent change its outgoing edges by applying its neighbor selection policies. The initial ring structure in Watts' experiments guarantees that the graph starts with a high clustering coefficient. In our case, if we break down clustering into pieces, such as interest clustering, specialty clustering, and so on, then the ring structure may not guarantee high interest clustering.

Adamic *et al.* [2001] study different local search algorithms in power-law networks to exploit the advantages of having nodes with high out-degrees. The variations on their local search strategies is similar to our choice of 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. In their approach, a peer with high out-degree is chosen because it will allow the message to get to more peers. In our case, we are not concerned about maximizing the number of agents (on the contrary) but trying to ensure 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.

Foner [1997] develops a matchmaking system, Yenta, to find (and make introductions between) people with similar interests. Yenta is based on building clusters of similar people using local knowledge. Once these clusters are built, then each person can query its own cluster to find answers. The implicit assumption here is that there will exist an expert in the cluster of people with similar interests. For this reason, the simulation results consider factors that increase the speed of clustering.

Wang develops an approach for organizing agents into communities based on the similarity of their interests and expertise [2002]. 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. In both Foner's approach and Wang's approach, it is assumed that clustering improves the efficiency of locating agents. When the agents' interests and expertise are more diverse, we believe that our observation of clustering does not favor quality, will dominate.

Ng *et al.* [2001] study the stability of PageRank and HITS algorithms and propose variants of these algorithms that offer more stability with respect to the addition or deletion of vertices. They take a graph-theoretic stance and simply recalculate the given metrics. Their results are of course valuable in judging the stability of the different algorithms. By contrast, however, we consider the processes explicitly. Thus we can address the question of how the links will evolve.

Several recent approaches try to improve the current PageRank algorithm. Haveliwala [2002] develops an approach where the PageRanks of the pages are influenced by the current query. Instead of generating one PageRank vector, several PageRank vectors are generated based on a predefined set of topics. Then, the query is mapped to one of the topics and the related PageRank vector is used to compute the rankings. In some respects, our approach exhibits a similar intuition in that neighbors are contacted and referrals are given in a highly topic-sensitive manner. However, we consider the PageRank computations based on the neighborhood relation—an agent's neighbors are the acquaintances that it values the most, all things considered.

Overall, we believe that a process-centric view as we discussed in Chapter 4 does not detract from the previous studies, but adds additional depth and understanding of how networks evolve in cooperative settings, but under local control. It also suggests how to apply various policies to maximize the quality of a given network of agents. This is of an immediate value for smaller networks, (for instance, for knowledge management by locating experts [Yu and Singh, 2003]), and of a longer-term value in understanding the evolving Web at large.

6.1.3 Trust

Lattice-based access control models have been used in computer security to regulate information flow [Sandhu, 1993]. Each node in the lattice denotes a different set of security privileges, called security classes. The more sensitive security classes are placed higher in the lattice. The flow of information is only allowed from the lower security classes to the higher ones. Thus, even though the less confidential information from lower security classes can be carried to the upper security classes, no confidential information flows down. This is similar to how we handle service types. Providers that can perform services higher up in the service graph can also perform lower services. In addition, we promote providers from lower service types to higher ones based on the providers' performance on the lower services.

Wille [2001] uses concept lattices for knowledge discovery in databases. The data objects are classified into meaningful concepts based on common attributes. The concepts then are arranged in a line diagram, which represents the concepts and the subconcept relationships among concepts. This representation is a structured way to visualize and analyze information.

McDonald [2001] develops a centralized expertise recommender system (ER) to be used in a computer supported collaborative environment. Each user enters a topic area and keywords to find an expert in the system. ER then returns a set of contacts who have expertise in these areas. Different heuristics are used based on the domain. The two heuristics are for technical development and technical support. The heuristic for technical development identifies the last person who has made changes to the specified code. The heuristic for technical support finds people who have dealt with similar problems before, based on the logs.

McDonald's approach differs from our approach in a number of ways. First, our approach is decentralized in that each agent searches the network itself (with help from its neighbors) without needing a centralized server. Second, we allow queries to span multiple domains (the dimension of the interest vector) and do not employ domain-based heuristics. In a system where the number of domains is large, finding domain-specific heuristics is hard. Further, there is no established method to combine these heuristics.

Yu and Singh [2002b] develop an approach for distributed reputation management where the reputation of an agent is computed based on testimonies of the witnesses using the Dempster-Shafer theory of evidence. They show how this model can be used to detect agents who are non-cooperative or agents who abuse their reputation by slowly decreasing their level of cooperativeness. Since the witnesses are found through referrals, Yu and Singh's approach captures social trust. Local evaluations are captured through belief functions, but relationships among service types are not captured. In our approach, we can capture relationships among service types using service graphs.

Barber and Kim [2001] propose an approach wherein agents use a belief revision algorithm to combine evidence they receive from other agents. In addition to providing evidence, each agent specifies its level of confidence in the evidence. Barber and Kim's approach captures social trust, but contrary to our approach, the trustworthiness of agents who provide evidence are not considered. Their approach does not consider local evidence, i.e., the previous interactions of the trustor with the trustee. In our approach, it is easier to handle local evidence as local policies of individual agents.

Pujol *et al.* [2002] calculate the reputation of an agent based on its position in its social network. The social networks are built based on the link structure induced by the Web pages of the users. An agent gets a high reputation only if the agents that point to it also have high reputation, similar to the notion of authority exploited in search engines such as Google. Pujol *et al.* test their approach to find the reputations of authors where the reputation of an author is defined as the number of citations received. Even though each agent can calculate its own reputation based only on local information (i.e., the agents that point at it), a central server is needed to access others' reputations. This approach does not capture local trust, since direct interactions are not taken into account. It captures social trust since the reputation of an agent is derived through how other agents have linked to it, but has no means to correct that based on local observations of an agent. In other words, the link structure is static and the positions of the agents do not change based on their interactions. In our approach, we allow agents to change neighbors using the neighbors' ability to give referrals as a heuristic. This allows us to rate the sources.

Sabater and Sierra [2002] develop a system for reputation management where reputations are derived based on direct interactions as well as the social relations of the agents. They use the number of interactions and the variance in ratings to derive the trustworthiness of the agent through direct interactions. To assess the trustworthiness through indirect interactions, Sabater and Sierra use fuzzy inference to combine evidence from multiple witnesses. In this regard, their approach captures both social and local trust. On the other hand, Sabater and Sierra do not offer a mechanism to propagate trust across related services as we have done in this dissertation.

Currently, we propagate trust based on a provider's trustworthiness for a single service.

However, sometimes it would help to combine the trustworthiness of the provider in several services. For example, if a service is composed of several smaller services, the trustworthiness of the provider in all the subservices will affect the trustworthiness of the provider in the composed service. This problem is also acknowledged by Sabater and Sierra [2002].

Sen and Sajja [2002] develop a reputation-based trust model used for selecting processor agents for processor tasks. Similar to our notion of service providers, each processor agent can offer varying performance. Agents are looking for trustworthy processor agents to interact with using only evidence from their peers. Sen and Sajja propose a probabilistic algorithm to find the number of agents to query to guarantee finding a trustworthy party. In our framework, we model the peers based on their prior performance and choose whom to ask for help based on these models. Thus, agents also decide the trustworthiness of the information source. However, in Sen and Sajja's framework, these models are not captured. All peers are treated the same, independent of their previous behavior. This approach does not handle local trust, since previous interactions of an agent with processor agents are not taken into account.

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

6.2 Directions

We describe two directions for further research.

6.2.1 Incentives

Creating incentives for answering queries could help to model the system more realistically. Recall that with the default answering policy an agent tries to answer each incoming question. If it cannot answer a question, it refers neighbors as dictated by the referral policies. In real life, we would expect different answering policies to be employed. At one extreme, an agent might try to answer all questions in order to aggressively disseminate its views. The agents who generate inaccurate answers or referrals are inherently punished since the other agents who receive these wrong answers will eventually decrease the expertise or sociability of that agent. In other words, if an agent talks too much without enough expertise, it will start getting fewer questions. Thus, the system discourages this type of an answering policy. At the other extreme, an agent may not answer questions since there might not be any incentives for providing services or referrals. Hence, it is possible to not provide any services to anybody but still receive good quality services. This is a common problem that plagues many existing peer-to-peer systems as well. A recent study on a partial trace of Gnutella network showed that 70% of the users on the network are *free-riders*. That is, they do not share any files themselves but use the system to download files from others [Adar and Huberman, 2000]. Obviously this is not desirable.

One way to cope with this problem is to model the network as a market where there is a benefit in answering questions and referring the right agents as well as a cost for asking questions. Each agent in this model is a utility-maximizer. Thus, in order to ask questions, each agent has to also answer questions or make referrals. Market-oriented approaches have been applied before to create incentives for sharing [Golle et al., 2001]. A second approach is based on reciprocity. In its tightest definition, a relation between two agents is *reciprocal* if each agent considers the other a neighbor. The role of reciprocity in solving the incentives problem is intuitively plausible. If an agent does not help others, it will not be helped. Reciprocity has been used in different disciplines like multiagent systems [Sen, 1996], anthropology, and social networks [Scott, 1991] to increase the cooperation among agents. A broader definition of reciprocity is based on an agent helping other agents in the group in which it is involved. Recent work in multiagent systems based on this broader definition of reciprocity involves studying the evolution of social rationality among agents [Hales and Edmonds, 2003]. It would be interesting to study incentives in our framework, especially its effect on the quality and the structure of the network.

6.2.2 Caching

In this dissertation, we have focused on services that can only be provided by some specified service providers. On the other hand, many information services can easily be cached from service providers and served to others. We have developed a caching technique, Marmara, that works with our referrals-based framework [Yolum and Singh, 2002]. In Marmara, cache entries are coupled with metadata, thereby allowing the use of heuristics and flexible queries for more informed searches. The entries that are of interest to more agents are replicated at more peers, providing on-demand performance improvement and fault tolerance. Our initial results show that when caching is employed, agents find good answers more easily [Udupi et al., 2003]. We defer deeper studies of these topics to future research.

6.3 Design Rules

Building applications of referral systems requires many design decisions. The properties identified through simulations as done in this dissertation can be used as design rules for real-life applications of referral systems. Here, we outline some possible usage of the properties observed in this dissertation.

As shown in Chapter 3, neither too many referrals nor too few referrals create high quality referral networks (Observations 2 and 3). 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 undesirable network structures have been identified (Observations 5 and 6). 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.

Chapter 4 shows that using certain agent strategies, the in-degree distribution and the PageRank distributions can exhibit a power law. Depending on the application, power law distributions may be desired. A power law distribution of PageRanks shows that agents actually identify others as authoritative. That is, actually useful agents are identified and used by the others to find information. This is certainly important and could be enforced in a referral system. Conversely, for example, for a knowledge management domain, a power law distribution 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 to avoid power law distribution of in-degrees.

A referral system should be robust enough to handle the problem of boot-strapping, i.e., enabling new agents to find parties to interact with as well as informing others of newcomers. For example, in the current simulations, if an agent is initially not pointed to by others, it cannot be located afterwards. A referral system can employ heuristics to ensure that agents become aware of each other.

As shown in Chapter 5 through Observation 19, service graphs can efficiently and effectively applied in conjunction with referrals. Again, the observed trade-offs between promotion error and effectiveness can be incorporated into a real system to ensure that agents only try a few reliable service providers that they trust for a new service.

The identification of such properties of referral networks brings us closer to enabling referral systems that can efficiently and effectively operate in open and dynamic environments.

Bibliography

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

Lada A. Adamic and Bernardo A. Huberman. Power-law distribution of the World Wide Web. *Science*, 287(5461):2115, March 24 2000.

Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Physics Review E*, 64(46135), 2001.

Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000. www.firstmonday.dk.

Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the World-Wide Web. *Nature*, 401:130–131, September 1999.

Robert L. Axtell. Zipf distribution of U.S. firm sizes. *Science*, 293:1818–1820, 7 September 2001.

Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: The topology of the World Wide Web. *Physica A*, 281:69–77, 2000.

K. Suzanne Barber and Joonoo Kim. Belief revision process based on trust: Agents evaluating reputation of information sources. In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, LNAI 2246, pages 73–82. Springer-Verlag, 2001.

Ronald Bonnell, Michael Huhns, Larry Stephens, and Uttam Mukhopadhyay. MINDS: Multiple intelligent node document servers. In *Proceedings of the 1st IEEE International Conference on Office Automation*, pages 125–136, 1984.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

Andrei Broder, Ravi Kumar, Farzin Maghou, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the Web. *Computer Networks*, 33(1–6):309–320, 2000.

Cristiano Castelfranchi and Yao-Hua Tan. The role of trust and deception in virtual societies. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS 34)*, volume 7, pages 7011–7018. IEEE Computer Society Press, January 2001.

Frank Dabrek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the ACM Symposium on Operating System Principles (SOSP)*, pages 202–215, 2001.

Partha Dasgupta. Trust as a commodity. In *Gambetta [2000]*, chapter 4, pages 49–72. 2000.

Chris Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 150–157. ACM Press, October 2000.

Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM Press, 2001.

eBay.com, 2003. http://www.ebay.com.

Rino Falcone and Cristiano Castelfranchi. The socio-cognitive dynamics of trust: Does trust create trust? In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber-societies*, LNAI 2246, pages 55–72. Springer-Verlag, 2001.

Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power law relationships of the Internet topology. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 251–262. ACM, 1999.

Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of Symposium on Discrete Algorithms (SODA)*, pages 94–103, 2002.

Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of Web communities. *IEEE Computer*, 35(3):66–70, March 2002.

Lenny Foner. Yenta: A multi-agent, referral-based matchmaking system. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 301–307, 1997.

Freenet. Home page, 2001. http://freenet.sourceforge.net.

Diego Gambetta, editor. *Trust: Making and breaking cooperative relations*. Electronic edition, 2000. Available at http://www.sociology.ox.ac.uk/papers/trustbook.html.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

Gnutella. Home page, 2001. http://gnutella.wego.com.

Philippe Golle, Kevin Leyton-Brown, and Ilya Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 3rd International Conference on Electronic Commerce (EC)*, pages 264–267, 2001.

Tyrone Grandison and Morris Sloman. A survey of trust in Internet applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16, December 2000.

David Hales and Bruce Edmonds. Evolving social rationality for MAS using "tags". In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, July 2003. To appear.

Taher H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference (WWW11)*, pages 517–526. ACM Press, May 2002.

Dennis E. Hinkle, William Wiersma, and Stephen G. Jurs. *Applied Statistics for the Behavioral Sciences*. Houghton Mifflin, Boston, 1979.

Bernardo A. Huberman. *The Laws of the Web: Patterns in the Ecology of Information*. MIT Press, Cambridge, MA, 2001.

Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. In *Huhns and Singh [1998b]*, chapter 1, pages 1–23. 1998a.

Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francisco, 1998b.

Catholijn M. Jonker and Jan Treur. Formal analysis of models for the dynamics of trust based on experiences. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW)*, LNAI 1647, pages 221–232. Springer Verlag, 1999.

Lalana Kagal, Tim Finin, and Anupam Joshi. Trust-based security in pervasive computing environments. *IEEE Computer*, 34(12):154–157, December 2001.

Gene Kan. Gnutella. In *Oram [2001]*, chapter 8, pages 94–122. 2001.

Henry Kautz, Bart Selman, and Mehul Shah. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.

Maurice G. Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251, November 1945.

Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201, November 2000.

Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. In *Proceedings of the Eighth World Wide Web Conference*, pages 1481–1493, 1999.

Adam Langley. Freenet. In *Oram [2001]*, chapter 9, pages 123–132. 2001.

Niklas Luhmann. Familiarity, confidence, trust: Problems and alternatives. In *Gambetta [2000]*, chapter 6, pages 94–107. 2000.

Matteo Marsili and Yi-Cheng Zhang. Interacting individuals leading to Zipf's law. *Physical Review Letters*, 80(12):2741–2735, 1998.

David W. McDonald. Evaluating expertise recommendations. In *Proceedings of the ACM 2001 International Conference on Supporting Group Work*, pages 214–223, 2001.

Barbara A. Misztal. *Trust in Modern Societies*. Blackwell Publishers, Cambridge, MA, 1996.

Wentao Mo. A referral-based recommender system for e-commerce. Master's thesis, Department of Computer Science, North Carolina State University, Raleigh, 2001.

Bonnie A. Nardi, Steve Whittaker, and Heinrich Schwarz. It's not what you know, it's who you know: Work in the information age. *First Monday*, 5(5), May 2000.

Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors and stability. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 903–910, 2001.

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

Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using PageRank to characterize Web structure. In *8th Annual International Computing and Combinatorics Conference (COCOON)*, pages 330–339, 2002.

David Pennock, Gary Flake, Steve Lawrence, Eric Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the Web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211, April 2002.

C. Greg Plaxton, Rajmohan Rajaraman, and Adrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the Ninth Annual Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

Josep M. Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 467–474. ACM Press, July 2002.

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott 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.

Sean Rhea, Chris Wells, Patrick Eaton, Dennis Geels, Ben Zhao, Hakim Weatherspoon, and John Kubiatowicz. Maintenance-free global data storage. *IEEE Internet Computing*, pages 40–49, September-October 2001.

Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001a.

Antony Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the ACM Symposium on Operating System Principles (SOSP)*, pages 188–201, Banff, Canada, October 2001b.

Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 475–482. ACM Press, July 2002.

Gerard Salton and Michael J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

John Scott. *Social Network Analysis: A Handbook.* Sage Publications, London, 1991.

Sandip Sen. Reciprocity: A foundational principle for promoting cooperative behavior among self-interested agents. In *Proceedings of the 2nd International Conference on Multiagent Systems*, pages 322–329. AAAI Press, Menlo Park, 1996.

Sandip Sen and Neelima Sajja. Robustness of reputation-based trust: Boolean case. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 288–293. ACM Press, July 2002.

Munindar P. Singh. The future of agent communication. In Marc-Philippe Huget, editor, *Communication in Multiagent Systems: Background, Current Trends and Future*. Springer Verlag, 2003. To appear.

Munindar P. Singh, Bin Yu, and Mahadevan Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, April 2001.

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari 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.

Piotr Sztompka. *Trust: A Sociological Theory*. Cambridge University Press, UK, 1999.

Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1996.

Yathiraj B. Udupi, Pınar Yolum, and Munindar 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)*, 2003. To appear.

Ronald E. Walpole and Raymond H. Myers. *Probability and Statistics for Engineers and Scientists*, chapter 8. Macmillan, New York, 2nd edition, 1978.

Fang 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.

Stanley Wasserman and Katherine Faust. *Social Network Analysis*. Cambridge University Press, New York, 1994.

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

Douglas Brent West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2001.

Rudolf Wille. Why can concept lattices support knowledge discovery in databases? In *Proceedings of the International Workshop on Lattice-based theory, methods and tools for Knowledge Discovery in Databases*, pages 7–20, 2001.

Hao Chi Wong and Katia Sycara. Adding security and trust to multi-agent systems. In *Proceedings of Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies*, pages 149–161, May 1999.

Pınar Yolum and Munindar 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.

Pınar Yolum and Munindar P. Singh. Emergent properties of referral systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. ACM Press, July 2003a. To appear.

Pınar Yolum and Munindar P. Singh. Personalized emergent communities in referral networks. In *Proceedings of the IJCAI Workshop on Intelligent Techniques for Web Personalization*, August 2003b. To appear.

Bin Yu. *Emergence and Evolution of Agent-based Referral Networks*. PhD thesis, Department of Computer Science, North Carolina State University, Raleigh, 2001.

Bin Yu and Munindar P. Singh. An agent-based approach to knowledge management. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 642–644, 2002a. Poster.

Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management.

In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 294–301. ACM Press, July 2002b.

Bin Yu and Munindar 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.

Bin Yu, Mahadevan Venkatraman, and Munindar P. Singh. An adaptive social network for information access: Theoretical and experimental results. *Applied Artificial Intelligence*, 2003. In press.

Ning Zhong, Jiming Liu, and Yi Yu Yao. In search of the wisdom Web. *IEEE Computer*, 35(11):27–31, November 2002.

G. K. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA, 1949.

Appendix A

List of Metrics

Capability:

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

Similarity:

$$I_A \oplus I_B = \frac{e^{-||I_A - I_B||^2} - e^{-n}}{1 - e^{-n}} \tag{A.2}$$

PageRank:

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

Sociability Rank:

$$S(i) = d \sum_{j \in I_i} (S(j) * \frac{\sigma_{j,i}}{\sum_{k \in N_j} \sigma_{j,k}}) + (1 - d) \tag{A.4}$$

Quality:

$$\frac{I_A \otimes E_B}{path(A, B)} \tag{A.5}$$

Spearman's Correlation:

$$\rho(C, D) = 1 - \frac{6 \sum_{i=1}^{n} (C_i - D_i)^2}{n(n^2 - 1)} \tag{A.6}$$

Interest Clustering:

$$\gamma(i) = \frac{\sum_{(u,v) \in M_i} I_u \oplus I_v}{|V_i|(|V_i| - 1)} \tag{A.7}$$

Appendix B

Simulation Parameters

The following parameters are used to configure the simulation test bed.

- TOTAL_NUMBER_OF_AGENTS specifies the number of agents in a given simulation. The default value is 400.
- NUMBER_OF_DIMENSIONS specifies the number of dimensions the interest and the expertise vectors have. For the cases where sociability is represented as a vector, sociability vector also has this many dimensions. The default value is 4.
- PERCENTAGE_OF_EXPERTS specifies the percentage of experts in at least one domain. By default, each expert has high expertise in exactly one domain, and there are an equal number of experts for each domain. The default value is 5%.
- MAXIMUM_PERTURBATION_PERCENTAGE_FOR_GENERATING_QUERY defines the maximum amount the interest vector will be perturbed to generate a query. The default value is 0.10.

- `MAXIMUM_PERTURBATION_PERCENTAGE_FOR_GENERATING_ANSWER` defines the maximum amount the expertise vector will be perturbed to generate an answer. The default value is 0, which means that the expertise vector of the answering agent will constitute the answer.
- β is the learning coefficient which is used to update the sociability of an agent i that has given a referral. For a referral that leads to a good answer, the sociability of the agent is increased as in Equation B.1 [Yu et al., 2003]. The default value for β is 0.20. Here, τ denotes the length of the referral path starting from the agent that poses the query to the agent that answers the query. ι denotes the distance of agent i from the query originator. Two agents may have both given referrals that help to eventually locate a service provider. Between these two agents, the sociability of the agent that is closer to the service provider is increased more.

$$S_i = S_i + \frac{(1 - S_i)\beta}{\tau - \iota} \tag{B.1}$$

If the agent i gives a referral that leads to a bad answer, then the sociability of agent i is decreased as shown in Equation B.2 [Yu et al., 2003].

$$S_i = S_i - \frac{S_i\beta}{\tau - \iota} \tag{B.2}$$

- ϕ is the learning coefficient which is used to update the expertise of an answering agent. When agent i returns an answer A_i to an agent j, agent j uses the following formula to update i's expertise.

$$E_i = (1 - \phi)E_i + \phi A_i \tag{B.3}$$

That is, we assume the answer A_i represents the expertise of agent i and hence use this value to adjust the estimated expertise of agent i. If the given answer is high, then the expertise of agent j increases. If the given answer is low, then the expertise of the agent decreases. The default value for ϕ is 0.25.

- `PERCENTAGE_DECAY_FOR_NO_ANSWER` (δ) is a coefficient used to penalize an agent that does not answer a query. The default value is 0.10. The expertise of the agent is decreased to $(1 - \delta)$ times its actual value.
- `SIMILARITY_THRESHOLD_FOR_ASKING_NEIGHBOR` is the threshold that determines whether a neighbor will be contacted for a query. After the capability metric is applied to the query and the modeled expertise of the neighbor, it is compared to this threshold. If the value exceeds the threshold, then the neighbor is contacted. The default value is 0.01. Having a small value ensures that at least most of the neighbors will be contacted.
- `SIMILARITY_THRESHOLD_FOR_ANSWERING_QUESTIONER` is the threshold that determines whether an agent should answer a query it receives. The agent applies the capability metric to the incoming query and its own expertise vector. If the resulting value is above the threshold, then the agent answers the query. The default value for this threshold is 0.3. This value ensures that only agents with high expertise attempt to answer the query.
- `SIMILARITY_THRESHOLD_FOR_REFERRING` is the threshold used by the *Refer all* and *Refer all matching* policies. When the threshold is taken to be 0.1, the *Refer all* policy is in effect. Other thresholds apply to the *Refer all matching* policy. The default referral policy is *Refer all matching* with the `SIMILARITY_THRESHOLD_FOR_REFERRING` set to 0.2.
- W denotes the weight of sociability in choosing neighbors. The neighbor selection policies use this weight. When W is set to 0, only the expertise of the

neighbors is considered. When W is set to 1, only sociability of the neighbors is considered. The default neighbor selection policy is *Weighted average*, where W is set to 0.5.

- `SIMILARITY_THRESHOLD_FOR_EVALUATING_ANSWER` is a threshold used for evaluating an answer. When an agent receives an answer to its query, it applies the capability metric to the query and the answer. If this value is above the threshold, then the answer is considered useful. The default value is 0.2.
- `NUM_NEIGHBOR_SET_CHANGES` denotes how many times each agent can change its neighbors. By default experiments are run until the agents stop changing their neighbors, i.e., until each agent is satisfied with its neighbors.
- `NUM_QUESTIONS_PER_NEIGHBOR_SET` denotes how many queries are allowed before a neighbor change. The default value is 3.
- `WAITING_LIMIT_FOR_MESSAGES` denotes the time-out after which the agents stop accepting answers and generate a new query. The default value is $7,000$ milliseconds.
- `MAXIMUM_REFERRAL_CHAIN_LENGTH` denotes the maximum number of referrals that will be followed on one path. The default value is 3.

Appendix C

Simulation Experiments

Here, we report additional details of the experiments. The results reported in the experiments are the average of three runs with different random number generator seeds.

- Community Comparisons: The experiments shown in Figures 3.3 and 3.5 compare communities that are drawn from different populations that use neighbor selection policies of *Providers, Weighted average* or *Sociables.* In all cases, the `NUM_NEIGHBOR_SET_CHANGES` is set to 20. The remaining values are set to their default.
- Efficiency and Effectiveness: The experiments shown in Figures 3.6 and 3.7 compare referral policies in terms of effectiveness and efficiency, respectively. The figures show the cases where each agent has 8 neighbors for 10, 20, and 25 percent providers. The `MAXIMUM_REFERRAL_CHAIN_LENGTH` for the shown experiments is 5 and the simulations are terminated after agents stop changing neighbors. With this setup, agents change neighbors no more than 4 times. The experiments have been repeated for the following case: Each agent has 4 neighbors instead of 8 and the `MAXIMUM_REFERRAL_CHAIN_LENGTH` is set to 3. The number of neighbor changes needed for the system to stabilize

is then 7 neighbor changes. The results of the experiments are similar.

- Clustering: In the clustering experiments, each agent has 4 neighbors and the experiments are run for 20 neighbor changes. Since there is no change after the 14th neighbor change, Figure 3.12 plots the first 14 neighbor changes. The experiments use the *Refer all matching* with threshold 0.2 as the referral policy and compare different neighbor selection policies. Figure 3.13 uses the same setup but only plots the first 4 neighbor changes, since that is enough to illustrate the possible inverse relation between quality and clustering.
- In-degree Distributions: The reported in-degree distribution experiments use 5% experts, but have been repeated with 10% and 20% experts; the results are similar.
  - The experiment shown in Figure 4.2 uses the *Providers* neighbor selection policy and the *Refer all* referral policy.
  - The experiment shown in Figure 4.2 uses the *Providers* neighbor selection policy and *Refer all matching* referral policy with the threshold set to 0.2. A threshold of 0.3 yields similar results.
  - The experiment shown in Figure 4.5 uses the *Sociables* neighbor selection policy and *Refer all* referral policy.
  - The experiment shown in Figure 4.6 uses the *Sociables* neighbor selection policy and *Refer all matching* referral policy with the threshold set to 0.2. Higher thresholds yield similar results. Similarly, replacing the *Sociables* policy with *Weighted average*, with W set to 0.5, realizes the same distributions.
- PageRank Distributions: Using the default referral and neighbor selection policies, Figure 4.7 plots the PageRank distribution for 5%, 10%, and 20% experts.

The experiments are run until the PageRank distributions stabilize. The stabilization happens at most after the 10th neighbor change. Hence, these simulations are run for 10 neighbor changes. For each neighbor change, the PageRank calculations are iterated, until the difference between two consecutive PageRank iterations is not greater than 0.001 for any of the agents. The calculations of PageRanks using eigenvectors yield the same ranking but different PageRank values since the eigenvector method normalizes the PageRank values. Figure 4.8 plots the PageRank distribution for the three referral policies, using 5% experts. The remaining parameters are set to their default value. Figure 4.9 plots the PageRank distribution after every two neighbor changes for the *Weighted average* neighbor selection policy, using 5% experts. Figure 4.10 plots the PageRank distribution for different neighbor selection policies, using 5% experts. For the PageRank evolution experiments (Figures 4.11 and 4.12), the simulation is run with default parameters and policies. Then, the described perturbations are made.

- Service Graphs: The experiments discussed in Chapter 5 contain 100 agents, whose expertise and interest span 9 dimensions. Each agent has three neighbors. The remaining parameters are set to their default values. The same experiments have been repeated with 16 dimensions and 50 service providers, and the results are similar.

The simulation test bed is easily configurable to allow the study of other properties of referral networks. For example, a study on the learning aspects of referral networks could study the learning parameters β and ϕ to understand the effects of different rates of learning or even different learning algorithms.