Searching Social Networks[*] [†]

Bin Yu
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA
byu@eos.ncsu.edu

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

ABSTRACT

A referral system is a multiagent system whose member agents are capable of giving and following referrals. The specific cases of interest arise where each agent has a user. The agents cooperate by giving and taking referrals so each can better help its user locate relevant information. This use of referrals mimics human interactions and can potentially lead to greater effectiveness and efficiency than in single-agent systems.

Existing approaches consider what referrals may be given and treat the referring process simply as path search in a static graph. By contrast, the present approach understands referrals as arising in and influencing dynamic social networks, where the agents act autonomously based on local knowledge. This paper studies strategies using which agents may search dynamic social networks. It evaluates the proposed approach empirically for a community of AI scientists (partially derived from bibliographic data). Further, it presents a prototype system that assists users in finding other users in practical social networks.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Design, Experimentation, Human Factors

Keywords

social networks, referral systems, knowledge management

[*]This research was supported by IBM and by the National Science Foundation under grants IIS-9624425 (Career Award) and ITR-0081742. We are indebted to Joe Bigus, Zhengang Cheng, Mike Huhns, Mahadevan Venkatraman, and Pınar Yolum for useful discussions and comments, to Wentao Mo and Paul Palathingal for contributions to the implementation, and to Henry Kautz and Bart Selman for sharing their AI bibliographic data.

[†]A previous version of this paper appears in the AAMAS 2002 Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'03, July 14–18, 2003, Melbourne, Australia.
Copyright 2003 ACM 1-58113-683-8/03/0007 ...$5.00.

1. INTRODUCTION

Finding relevant information is a longstanding problem in computing. Conventional approaches such as databases, information retrieval systems, and Web search engines partially address this problem. Often, however, the most valuable information is not widely available and may not even be indexed or cataloged. Much of this information may only be accessed by asking the right people.

The challenge of finding relevant information then reduces to finding the right people whom we may ask a specific question and who will answer that question for us. The right people are those who have the desired information or expertise. Finding them involves naturally depends on our *social network:* our friends, our friends' friends, and so on. Clearly, building and maintaining a central repository of social relationships is not feasible: people usually cannot and, because of considerations such as privacy, will not list their social relationships in a repository.

For this reason, distributed search through referrals is more promising. Other researchers have noted the importance of referrals for human information flow [Brown and Reingen, 1987; Tassier and Menczer, 2001]. Further, there is evidence that referrals to acquaintances can be remarkably effective in searching large social networks. The sociologist Milgram discovered that strangers are connected via short chains of acquaintances [1967]. Milgram found an average of six links in his sample of pairs of strangers in the US—hence the pop culture concept of *Six Degrees of Separation*.

A *referral system* is a multiagent system in which the agents cooperate by giving, pursuing, and evaluating referrals. Each user is assigned an agent who learns the user's preferences and interests. The agent maintains a view of its user's acquaintances. Using these, the agent prioritizes incoming queries, often issuing referrals where others might be more suitable to field a given query. Based on its user's feedback, each agent rates those provided an answer and those who referred to them. Each agent modifies its neighbors accordingly. Consequently, the referral system evolves to reflect the part of the social network. We can think of a referral system as overlaying the social network of its users. Thus the problem of searching a social network reduces to routing queries in a corresponding referral system.

MINDS and ReferralWeb are two major approaches for referral systems. MINDS emphasizes learning heuristics for referral generation [Huhns et al., 1987], whereas ReferralWeb focuses on how to bootstrap the referral system [Kautz et al., 1997]. By contrast, this paper emphasizes the dynamics of social networks and the effects of the dynamics on information flow. Specifically, we consider how to efficiently search social networks with the help of agents who act only on the basis of local knowledge. We study referral systems empirically and show how to control search by adaptively choosing the referrals.

The study of referrals can support the development of multiagent systems that lack specialized agents such as brokers or facilitators [Decker et al., 1997] or which involve people and agents working with one another. Such multiagent systems apply in the following scenarios.

- *Knowledge Management*. Traditional approaches, which emphasize documents, miss out on the wealth of knowledge that is not indexed [Fischer and Ostwald, 2001]. Rapid organizational change further exacerbates the problem and increases the importance of social networks [Nardi et al., 2000]. Our approach helps to develop an effective, naturally occurring knowledge management system, in which the agents not only create and maintain the personal social networks of their users, but also help search and explore them.

- *Trust and Reputation Management*. In open systems, locating trustworthy and reputed parties, e.g., service providers, is crucial. Referrals enable agents to share information so that untrustworthy parties can be weeded out. We previously developed a probabilistic model of reputation in which an agent combines evidence from a number of witnesses regarding a particular party [Yu and Singh, 2002]. Referrals can be used to find reliable witnesses.

We have evaluated our approach in two main ways. We conducted simulation experiments seeded with an inferred community of about five thousand people to show how our approach leads to improved search. We also implemented a prototype system for knowledge management over a small real-life social network.

The rest of this paper is organized as follows. Section 2 provides an overview of referral systems. Section 3 describes our experimental results. Section 4 describes prototype referral system. Section 5 summarizes the relevant literature. Section 6 discusses the main themes and some directions for future research.

2. REFERRAL SYSTEMS

Intuitively, in a referral system, each agent helps its user maintain his personal social network. The nominal procedure is simple. A query from the user is seen by the agent, who suggests potential contacts to whom to send the query. After consultation with the user, the agent sends the query to the agents of the selected contacts. Each agent maintains a model of its user. An agent who receives a query can decide if it suits its user and, if so, forward it to the user. If not, the agent *may* respond with referrals to others. If the agent or user so wish, they can discard a query without responding to it. (An agent would not unilaterally discard a query, but would place it in a low-priority folder; however, let's assume that low priority queries are not looked at in time to have any bearing on the referral process or how the agent's learn about one another.)

A query specifies what information is being sought. A response, if given, includes an answer or a referral. An answer, if given, depends on the query and the expertise of the answering agent. An agent answers only if it is reasonably confident of its expertise matching the query. A referral depends on the query and on the referring agent's models of others; a referral is given only if the referring agent has sufficient confidence in the relevance of the agent being referred.

Each agent maintains models of its *acquaintances*. The closest acquaintances are called *neighbors*. An agent sends its query initially only to some of its neighbors. If an agent receives a referral, it may pursue it even if the referred party is not already an acquaintance—this is how acquaintances are added. An agent adapts its models of its acquaintances from its interactions with others, e.g., when they ask or answer a query. Each agent is allowed only a small number of neighbors; however, no limit is imposed on the number of acquaintances. Periodically, an agent may promote some of its acquaintances to becoming its neighbors and demote some existing neighbors to make room for the new ones.

When the originating agent receives referrals, it integrates them into its models. Based on its models, it may decide to actually follow up a referral. When the agent receives an answer, it uses the answer as a basis for evaluating the expertise of the agent who gave the answer. This evaluation affects its model of the expertise of the answering agent, and its models of any agent who gave a referral that led to this answering agent.

2.1 Modeling Expertise and Sociability

Each agent maintains two kinds of models: a *profile* for its user; and an *acquaintance model* for each of its acquaintances. We capture these models via the vector space model (VSM) [Salton and McGill, 1983], a classical information retrieval technique. The vectors in VSM are term vectors indicating a weight for each term. We adapt VSM to locate people rather than documents. In our formulation, the terms correspond to different areas of expertise. The expertise of each user is modeled as a term vector. Similarly, the query is modeled as a term vector.

In VSM, the similarity between two term vectors is defined as the cosine of the angle between them. We define the similarity between a query and an expertise vector as the cosine of the angle between them, but scaled by the length of the expertise vector. Intuitively, for two agents with expertise in the same direction, the one with the greater expertise is more desirable, whereas the traditional definition would treat them alike.

DEFINITION 1. *Given a query vector $Q = \langle q_1, q_2, \ldots, q_n \rangle$ and an expertise vector $E = \langle e_1, e_2, \ldots, e_n \rangle$, the similarity between Q and E is defined as:*

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

For example, consider a query vector $Q = \langle 0.1, 0.9 \rangle$ and two expertise vectors $E_1 = \langle 0.5, 0.5 \rangle$ and $E_2 = \langle 1, 1 \rangle$. In VSM, E_1 and E_2 are equally similar with the query vector Q, but in our approach, E_2 is better than E_1, since $Q \diamond E_2 > Q \diamond E_1$.

When an agent receives a query, it matches the query against the expertise vector in its user's profile. If there is a good enough match, the query is passed on to its user.

DEFINITION 2. *Given a threshold ω_i (where $0 \leq \omega_i \leq 1$), there is a match between user P_i and query vector Q if $Q \triangle P_i \geq \omega_i$.*

The sociability of an agent reflects its ability to give good referrals. Each agent evaluates others based on a linear combination of their expertise and sociability. That is, the relevance of a neighbor to a given query depends not only on the similarity of the query to the user's expertise, but also on the weight assigned to sociability versus expertise.

DEFINITION 3. *The relevance of a query vector Q to P_j is computed as $Q \triangle P_j = (1 - \eta)(Q \diamond E_j) + \eta S_j$, where E_j is the expertise of P_j, S_j is the sociability of P_j, and η is the weight given to sociability.*

Further, the user P_i may specify an absolute relevance threshold Ω_i. The threshold can be adjusted to tune the number of purported experts found and to limit the number of referrals that user P_i will give to other users. Note that usually we have $\Omega_i \geq \omega_i$.

DEFINITION 4. *Given a query vector Q (from the user P_i himself or another user) and a threshold Ω_i, a neighbor P_j of user P_i is relevant to Q if and only if $Q \triangle P_j \geq \Omega_i$ for a special value of η.*

Our previous work studied the effects of η on the quality of referral systems [Yu et al., 2003]. We found that a certain emphasis (during learning and querying) on the agents' referring ability improves the quality of the system, but that an overemphasis on referrals at the cost of expertise is not useful. For simplicity, we only consider the case $\eta = 0.3$ here.

2.2 Referral Graphs

Each agent learns its user's profile and its acquaintance models based on an evaluation of the answers received as well as the referrals that led to them. A *referral graph*, which is local to each agent, encodes how the computation spreads as a query originates from an agent and referrals or answers are sent back to this agent.

DEFINITION 5. *A referral r to A_j returned from A_i is written as $\langle A_i, A_j \rangle$, we say A_i is a parent of A_j and A_j is a child of A_i.*

For convenience, we include the initial query among the referrals. This enables us to write a referral chain of length l for a query originating with A_r as $\langle A_r, A_1, \ldots, A_l \rangle$. Then *ancestor* and *descendant* are easily defined based on parent and child, respectively.

The referral chains for a given query induce a directed graph whose root is the originating agent. The *depth* of a referral is its distance on the shortest path from the root. Our algorithms ensure that the graph remains acyclic.

DEFINITION 6. *A referral graph $G(Q)$ for a query Q is a rooted directed graph (A_r, Λ, R), where A_r is the requesting agent (root), $\Lambda = \{A_1, A_2, \ldots, A_n\}$ is a finite set of agents (vertices) that includes A_r, $R \subseteq \Lambda \times \Lambda$ is a set of referrals (edges).*

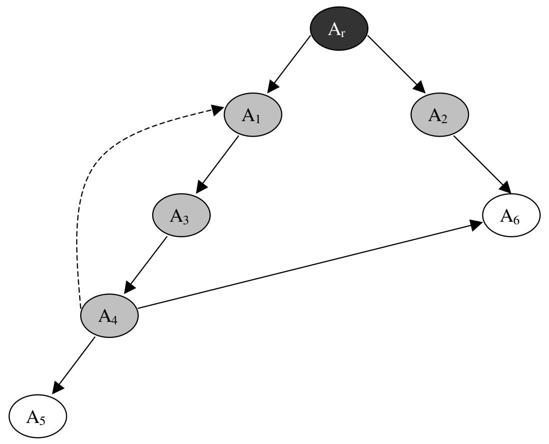

Figure 1: A referral graph generated from a query. The requesting agent is black; the agents that have been queried are gray; the agent who have not been queried are white.

DEFINITION 7. *A referral $r = \langle A_i, A_j \rangle$ is redundant for a referral graph (A_r, Λ, R), if and only if $A_i, A_j \in \Lambda$ and A_j is an ancestor of A_i with respect to R.*

Clearly, an acyclic referral graph includes no redundant referrals. In the context of Figure 1, a referral $\langle A_4, A_1 \rangle$ would be redundant, since A_1 is an ancestor of A_4. Referral $\langle A_4, A_2 \rangle$ is not redundant, since it introduces no cycles. Our construction algorithm avoids redundant referrals.

2.3 Weighted Referral Graph

Figure 1 shows a simple referral graph with two leaf agents A_5 and A_6. Which should the querying agent follow first? To support this decision, we introduce weighted referral graphs in which each agent (vertex) and referral (edge) are assigned a weight. The idea is that the agent with the greater weight is a better bet. Let w_i be the weight of A_i and w_{ij} be the weight of referral $\langle A_i, A_j \rangle$. This referral (to A_j) is given by A_i to the requesting agent A_r; now we assume A_i sends along w_{ij} as well.

DEFINITION 8. *A weighted referral graph $\overline{G}(Q)$ is a four-tuple (A_r, Λ, R, w), where (A_r, Λ, R) is a referral graph generated for query Q originating with agent A_r and w is the following assignment of weights to the vertices and edges of the given graph:*

- *Requesting agent (vertex A_r). $w_r = 1$.*
- *Referrals (edge from A_i to A_j). $w_{ij} = Q \triangle P_j$.*
- *Other agents (vertex A_j). $w_j = \sum_{\langle A_i, A_j \rangle \in R} w_i * w_{ij}$.*

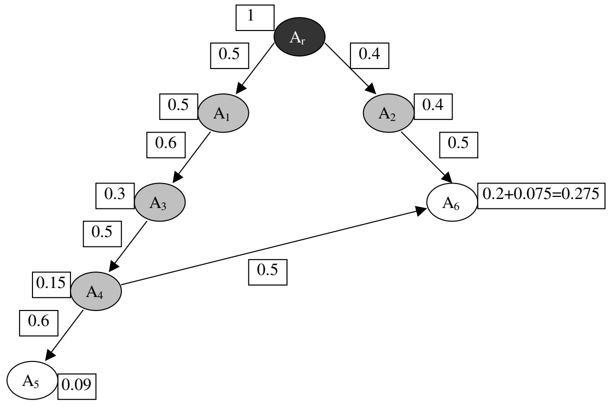

Figure 2: A weighted referral graph

Figure 2 shows an example of a weighted referral graph. Here, w_6 is $0.4*0.5+0.15*0.5 = 0.275$ and w_5 is 0.09. On adding a referral, we recompute the weights for the agents whose weight may have changed. Consider Figure 2. Suppose $\langle A_2, A_6 \rangle$ is received first. When we add $\langle A_4, A_6 \rangle$ to the referral graph, we recompute the weight of A_6.

DEFINITION 9. *Given $\overline{G}(Q) = (A_r, \Lambda, R, w)$, and a new referral $\langle A_i, A_j \rangle$, A_j is a cut-point of $\overline{G}(Q)$ if and only if $A_j \in \Lambda$.*

For example, in the above case, A_6 is the last agent on the referral chain $\langle A_r, A_2, A_6 \rangle$. A more interesting situation is if A_4 refers to A_2, where A_2 was already queried and referred to A_6 (here A_2 is a cut-point). In this case, we must propagate the changed weights to the descendants of A_2 via the operation *relax* applied to cut-points. The changed weights capture the fact that the referrals were, in effect, stronger than previously recorded in the graph.

Algorithm 1 Constructing a referral graph

```
 1: Suppose agent $A_r$ is the requesting agent, and $\Lambda$ is the set of
 2: agents being visited. For any referral $r = \langle A_i, A_j \rangle$, agent $A_r$
 3: will update the expertise and sociability of other agents
 4: according to the following rules
 5: if ($A_j \notin \Lambda$) and ($A_j$ returns an answer) then
 6:     Append $r$ to the referral graph
 7:     Add $A_j$ into $\Lambda$
 8:     Update the expertise of agent $A_j$ and the sociability of
 9:     any agent on the referral chain to agent $A_j$
10: else if ($A_j \notin \Lambda$) and ($A_j$ does not return an answer) then
11:     Append $r$ to the referral graph
12:     Add $A_j$ to $\Lambda$
13: else if ($A_j \in \Lambda$) and ($A_j \neq ancestor(A_i)$) then
14:     Append $r$ to the referral graph
15:     Relax $A_j$ and descendants of $A_j$
16: else
17:     Ignore referral $r$
18: end if
```

DEFINITION 10. *The operation relax(Agent A_i) updates (if necessary) the weight of A_i and of each of its children.*

Algorithm 1 presents the process of constructing a referral graph from a set of referrals. It considers the length of referral chains when expanding a leaf agent and prefers leaf agents with shorter referrals if their weights are the same. Note that the Agent IDs are the only global knowledge each agent has.

2.4 Propagating Rewards and Penalties

Now we discuss how the requesting agent updates its acquaintance models, i.e., assigns rewards and penalties, when an answer is received. A simple operator Π is used for updating the sociability and expertise of a given agent. The intuition behind Π is that sociability and expertise ratings should build up slowly, but fall quickly: that is, the agents are cautious in their dealings with others.

DEFINITION 11. *For X and Y, where $0 \leq X \leq 1$ and $-1 \leq Y \leq 1$, $\Pi(X, Y) = X + Y - XY$ if $0 \leq Y$, or $X + \frac{XY}{1+Y}$, otherwise.*

Given a referral graph G (or $\overline{G}(Q)$), suppose A_j returns an answer T. Then the requesting agent A_r will update the expertise and sociability of its models as follows, where α is the rating given by A_r's user, β is the learning rate, $-1 \leq \alpha \leq 1$, and $0 \leq \beta \leq 1$.

- *Expertise:* A_r will update the expertise vector for its own user as $(1-\beta)E_r + \beta Q$ and the expertise vector for A_j as $(1-\beta)E_j + \alpha\beta T$. A_j will update the expertise vector for P_j as $(1-\beta)E_j + \beta T$.

- *Sociability:* Suppose l is the depth of A_j in the referral graph. Algorithm 2 propagates credits (rewards or penalties) to A_j's ancestors according to their distance from A_j. The algorithm is invoked as *propagateCredits*$(A_j, l-1, \alpha)$.

The magnitude of the rewards or penalties is greater for agents who are closer to the answering agent. For example, in Figure 2, A_2 and A_4 give referrals to A_6. If A_6 returns an answer of quality α, A_2 and A_4 will get credit α, respectively. A_3 and A_1 will get credit $\alpha/2$. If there is no answer from A_j, there will be no penalties for the expertise of agent A_j and for the sociability of the intermediate agents on the referral chain.

Algorithm 2 Propagating credits or penalties in a referral graph

```
1: propagateCredits(Agent $A_i$, int l, double credits)
2: for each parent of $A_i$ do
3:     if ($l \geq 0$) and ($i \neq A_r$) then
4:         $A_j$ = parent($A_i$)
5:         $S_j = \Pi(S_j, \text{credits})$
6:         propagateCredits(parent($A_j$), l-1, credits/2)
7:     end if
8: end for
```

3. EXPERIMENTAL RESULTS

Networks of scientific collaborations have been studied recently by Newman [2001] and Barabási et al. [2002]. These works focus on the statistical properties of the networks, i.e., numbers of papers written by authors, numbers of authors per paper, typical distance from one scientist to another, the evolution over time of these qualities, and so on. By contrast, in our experiments we investigated the performance of expert location techniques, comparing a static network with an evolving network.

We reconstructed a social network for 4,933 AI scientists (each modeled as an agent) based on a bibliographic data corpus. The data is from the proceedings of AAAI (1980-2000) and IJCAI (1981-2001) conferences.[1] We extracted author, title, and keyword for each paper. We manually removed inconsistencies from author names (e.g., their spellings, abbreviations, ordering) and ensured that names are distinct if and only if the people named are distinct. Next we built the initial social network as follows.

- Using the keywords, we classified each paper into one of the nineteen topics in a taxonomy.[2] For example, the keyword *case-based reasoning* is mapped to *knowledge representation and reasoning*. The taxonomy is used only for categorizing papers; the agents do not model each other's expertise weights for the various topics.

- An author is considered another author's neighbor if they have coauthored one or more papers. To model social relationships that are not captured in the bibliographic data (e.g., if two friends never coauthored a paper in the selected proceedings), we introduce additional random links among the authors. The number of these links equals the number of links due to coauthorship.

- The expertise vectors in the models maintained by the various agents are initialized using the classical term-frequency inverse document frequency (TFIDF) approach [Salton and Buckley, 1988]. That is, each element e_i of an expertise vector $E = \{e_1, e_2, \ldots, e_n\}$, is derived by multiplying a term frequency (TF) component with an inverse document frequency (IDF) component. There are two cases.
  - In the profile maintained by A_j for its user, $e_k = \mathrm{tf}_j * \mathrm{idf}_k$, where tf_j is number of papers authored by P_j

[1]Available from http://dblp.uni-trier.de/db/conf/ijcai/ and http://www.aaai.org/Press/Proceedings/AAAI/.

[2]The 19 topics and corresponding numbers of papers are: AI architecture (224), agents and multiagent systems (265), applications (614), art and music (49), cognitive science (254), constraint satisfaction (271), expert systems (226), foundations (93), game playing (29), genetic algorithms (43), human-computer interaction (65), information retrieval (91), knowledge representation and reasoning (1692), logic programming (80), machine learning (806), natural language processing (549), neural networks (87), planning and search (537), and vision and robotics (660).

in topic k, and $\mathrm{idf}_k = \log(N/n_k)$, where N is the total number of papers (6,635), and n_k is the number of papers in topic k;

  – In the acquaintance model maintained by A_i for A_j, $e_k = \mathrm{tf}_j * \mathrm{idf}_k$, where idf_k is as above, but tf_j is defined as the number of papers coauthored by P_i and P_j in topic k.

- We identify the experts in the various topics. An author is an expert in a topic k if and only if the weight e_k is above a certain threshold. In our case, the threshold is set to 8, leading to 287 out of 4,933 authors being identified as experts. The feedback rating α is set to 1 if an expert is found. The other two thresholds ω_i (for filtering) and Ω_i (for referring) for each agent A_i are both set to 0.1. The sociability for all agents in acquaintance models is initialized to 0.5. The learning rate β is 0.1.

We consider queries corresponding to vectors of length 19 that are 1 in one dimension and 0 in all other dimensions. For example, $[1, 0, \dots, 0]$ would be a query in the topic of *AI architecture*. Typical authors have papers in one or two topics. Therefore, the queries can be distinguished into two categories: *home queries*, i.e., from a topic where the author has some papers; *foreign queries*, i.e., from topics where the author has no paper.

3.1 Effect of Branching Factors

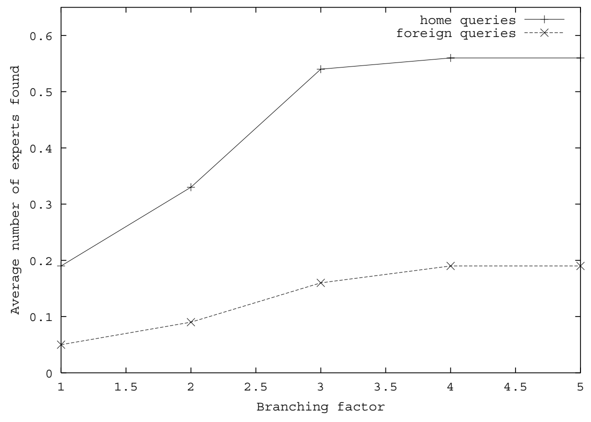

Figure 3: Average number of experts found for different branching factors

The first question we study is how many neighbors an agent should refer to while processing a query. Following Kautz et al., we term this the *branching factor* and denote it by F. The branching factor influences the number of experts that can be found with a given depth of referrals. Figure 3 shows the number of experts found (averaged over all agents) for different branching factors, while fixing the depth of referral graph as six. We find that $F = 3$ and $F = 4$ were needed to find all suitable experts for home and foreign queries, respectively. Below, we use $F = 4$ (unless otherwise specified) for both home and foreign queries. This is important because it suggests that referrals can support a focused search. That is, you can find the experts you need without spamming your friends and colleagues.

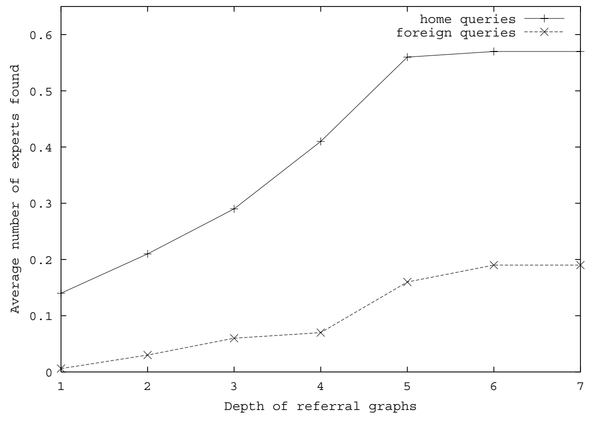

Figure 4: Average number of experts found for different depth of referral graphs

3.2 Depth of Referral Graphs

The next natural question is how deep would the referral graphs need to be for this network? We find that for home queries, the depth should be five, but for foreign queries, the depth should be six.

Figure 4 illustrates the power of referrals. Setting the maximum depth to one achieves the effect of contacting one's neighbors but not using referrals. For home queries, referrals yield a four-fold improvement in chance of finding an acceptable answer (from 14% to 57%). Referrals are even more important when seeking an expert in an area different from one's own. For foreign queries, referrals yield a thirty-fold improvement (from 0.6% to 19%). In each case the average number of experts found tend to level off after the depth is increased beyond a certain point. This indicates that the remaining experts are socially disconnected from the requesting agent.

3.3 Accuracy of Referral Chains

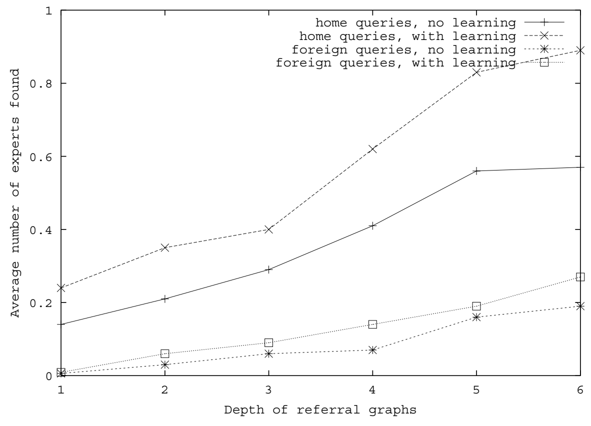

Figure 5: Average number of experts found in a dynamic referral systems

Everyone has only incomplete knowledge of his community. This is why social networks are useful in the first place. Some agents may not be good experts, but may be well connected and may give good referrals. In our approach, sociability credits the ability to give good referrals. The referring process considers both the expertise and the sociability of the different agents. The agents send queries, referrals, and responses to one another, all the while learning about each others' expertise and sociability. Note that the number of neighbors for each agent remains constant, but the set of neighbors is updated so that the most promising acquaintances are promoted to be neighbors. After each agent sends out ten home or foreign queries, we run the experiment again for different depths of referrals. Figure 5 shows that, even with only ten queries, the number of experts found can be significantly improved. This suggests that learning could be effective in practice, especially for home queries.

3.4 Minimizing Referral Graphs

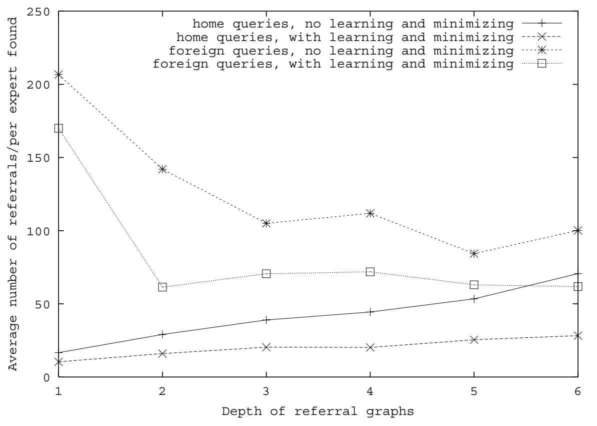

Figure 6: Average referrals per experts after minimizing referral graphs

Our last experiment is about how to minimize referral graphs by using weights as introduced in Section 2.3. When the requesting agent A_r receives referrals, it queries an agent with the highest weight. The referring process stops when an expert is found. Figure 6 summarizes the results when $F = 4$. We find that the average number of referrals per expert is significantly improved after minimizing the referral graph. This experiment indicates that the requesting agent can efficiently find short paths to the desired experts, even though the referrals are generated based on local knowledge.

4. A PROTOTYPE SYSTEM

The Multiagent Referral System, MARS, is a prototype system based on the above ideas. MARS agents give and take referrals as above. They also include an interface in which text queries can be entered by users.

A challenge for referral systems is how to bootstrap them. MARS uses a server where new users register themselves along with their topics of expertise and with which they can find existing users based on their self-stated interests. A unique UserID is assigned to the MARS agent when its user registers on the server. An agent may contact the registration server as a fall back mechanism if it cannot find a suitable contact on its own.

MARS is implemented in Java. It uses IBM's Agent Building and Learning Environment (ABLE) for its reasoner [Bigus et al., 2002]. The registration server is implemented over a Sybase DBMS. In the current version of MARS, MARS agents use email (using a dedicated server) as its transport mechanism. Due to the limitations of the current email server, and NCSU regulations, MARS was only evaluated by a small group of users at NCSU. Users can send queries in the topic of AI. MARS then helps search for the needed experts. Because of referrals, these experts need not be neighbors of the requesting party. We are replacing MARS' transport layer with Jabber, an open, XML-based protocol for instant messaging and presence (http://www.jabber.org).

5. RELATED WORK

Previous work on referral systems, peer-to-peer networks, and multiagent systems has also addressed the problem of searching large, complex networks. We review some of the key literature below.

5.1 Referral Systems

MINDS is a distributed information retrieval system, in which agents share both knowledge and tasks in order to cooperate in retrieving documents for users. Huhns et al. [1987] present a set of heuristics for learning and updating the relevance of documents to individual topics of interest. By dynamically learning document distribution patterns, as well as user interests and preferences, MINDS customizes document retrieval for each user.

Kautz et al. simulated expertise location in a large company and showed how the length and accuracy of referral chains are affected by the number of users, and the accuracy and responsiveness of each user [1996]. Kautz et al. developed ReferralWeb in which the co-occurrence of names in close proximity on Web pages is used to suggest direct person-to-person relationships. An early version used email logs to infer relationships between people, but later versions excluded email because of users' concerns over privacy.

Our work is similar in spirit to MINDS and ReferralWeb but extends them in two ways. MINDS incorporates learning, but does not explicitly use referral chains. ReferralWeb models a referral system statically as a graph and considers referrals directly through path search in the graph. However, it lacks a learning component for each agent, and cannot accommodate different strategies for choosing referrals for different queries.

ContactFinder is an agent that reads messages posted on bulletin boards, and extracts topic areas using a set of heuristics [Krulwich and Burkey, 1996]. It assists users by referring them to people who can help them. ContactFinder posts its referrals back to the bulletin boards, and hence that person's communication partners are not considered. The Knowwho email agent maps a user's social network by reading through his email messages [Kanfer et al., 1997]. Knowhow applies three techniques to improve the accuracy of referrals: (1) term-weighted document matching methods adapted to locating persons, (2) relevance feedback, and (3) semantic generalization for terms used in queries. It does not consider the sociability of each user, and strategies for controlling the referral process.

Vivacqua et al. develop a user-interface agent, called Expert Finder, which can assist a novice user in finding experts by matching the profiles of the novice and the expert [2000]. A MITRE project, also called Expert Finder, derives expertise estimation from newsletters, resumes, employee database and other information in an organization [Maybury et al., 2000]. MITRE's XperNet focuses on identification and tracking of expert communities using statistical clustering and network analysis. Answer Garden applies in help desks [McDonald and Ackerman, 2000]. It provides a branching network of diagnostic questions through which experts can navigate to match a novice's question. Answer Garden uses approximation techniques for mapping expertise networks (specializations of an organization's social network) within an organization. IKNOW assists users by generating referrals by searching an organization's databases [Contractor et al., 1989]. In other words, it answers questions such as *who knows what?* and *who knows who?* about the organization's knowledge network.

5.2 Peer-to-Peer Networks

The problem of searching a large network has been studied for peer-to-peer (P2P) networks. Typically, a P2P node broadcasts a search request to its peers, who propagate the request to their peers, and so on. By contrast, in a referral system, referrals are sent back to the requesting agent, who can adaptively direct or end the search.

Distributed search algorithms in Gnutella broadcast a request to all peers in a brute force manner. Chord [Stoica et al., 2001], CAN [Ratnasamy et al., 2001], and Pastry [Rowstron and Druschel, 2001] study distributed hashing, in which given an object, the algorithm will guarantee to locate a peer that has that object. In Chord, nodes are assigned a numerical identifier along a ring, while in CAN, nodes are a subrange of an N-dimensional torus. However, in these techniques, the routing table for each node is fixed and thus the network is not reconfigurable. Most importantly, the techniques are not applicable to social networks, which cannot be partitioned by IP address.

Yang et al. [2002] study performance and tradeoff of three search techniques: iterative deepening, directed BFS, and local indices. Directed BFS is similar to our approach, but in their approach, each node only maintains simple statistics for its neighbors Instead, in order to select neighbors more accurately, we model information about each neighbor in an expertise vector.

Adamic et al. [2001] and Kim et al. [2002] study the power-law of link distributions, and introduce a number of local search strategies that use high degree nodes. Such strategies may be helpful for people, who can decide to contact friends who are better connected than others, but they cannot readily be used in the design of agent-based referral systems. Our notion of *sociability* captures a similar intuition and enables each agent to learn about which of its neighbors are more effective at referrals.

The *small-world phenomenon* has been known for a long time [Milgram, 1967], but was not understood computationally until recently. Watts & Strogatz [1998] found that small-world networks are neither fully regular nor fully random. Such networks are highly clustered (like regular graphs) with just a few random short paths (like random graphs). Kleinberg [2000] found that it was only possible to find short paths for the model after randomly rewiring a two-dimensional lattice in a decentralized fashion. The topology of referral systems is similar to a two-dimensional lattice, but in our settings there is no global information about the position of the "target" agent. Hence, it is not possible to determine whether a move is toward or away from a target.

5.3 Matchmaking Systems

To be deployed in open settings, multiagent systems must provide effective, robust, and scalable mechanisms for locating agents. Classically, middle agents address this challenge [Decker et al., 1997]. Centralized architectures have a single middle agent, which provides location services to the other agents in the system [Decker et al., 1997]. Such architectures are simple to use, but do not scale well, have a single point of failure, and most importantly cannot offer multiple perspectives. Conventional distributed architectures use multiple middle agents, each with partial information of the system. The middle agents cooperate with one another to locate agents with desired services. Although such architectures can yield better performance and reliability than centralized architectures, they presuppose a fixed configuration. Therefore, these architectures are ill-suited to multiagent systems where agents can join and leave the system dynamically.

Shehory proposed a peer-to-peer location mechanism for open multiagent systems, in which each agent caches a list of agents it knows [1999]. Shehory's mechanism is similar to distributed search in peer-to-peer networks. Shehory studied the communication complexity of the above system based on lattice-like graphs, while we focus on how to efficiently find unknown agents in large and dynamic multiagent systems and social networks of unknown topology.

Matchmaking systems, such as SHADE [Kuokka and Harada, 1995] and Yenta [Foner, 1997], group or cluster users with similar interests. The basic idea behind matchmaking systems is bootstrapping each agent and finding at least one other agent with which to communicate and forming clusters of like-minded agents. When grouped together, users can easily find others with similar interests. However, matchmaking systems have no mechanism specifically for finding experts, so it is harder to find someone who has enough knowledge to help.

Collaborative filtering involves a server aggregating the choices of several users and making recommendations to a user based on the choices of users similar to the given user Schafer et al. [1999]. This approach has the limitation of identifying the user providing a rating to the server, while not revealing the source of recommendations. Our approach, by contrast, is decentralized and lets the users control to whom they reveal their ratings.

6. CONCLUSION

Social networks are a natural way for people to go about seeking information [Nardi et al., 2000]. Referral systems are promising because they capture two essential aspects of social networks: how they are applied and how they are evolved. In some applications, e.g., knowledge management, a referral system may only assist users in maintaining their social relationships; in other applications, e.g., trustworthy service location, the social relationships of interest may emerge among the agents.

The second class of applications relates to using a referral system as an ingredient of a practical multiagent system, where the brokerage and location services are handled through referrals. A referral system approach, being perfectly decentralized, would not only be more resistant to failure but would also lead to the dissemination of better vetted information, leading to superior performance across the system.

The above work has opened up some interesting avenues for further research. On the theoretical side, we plan to incorporate incentives and other mechanisms to encourage the participation of users and to discourage exploitation of helpful users. On the practical side, we plan to complete the transition of MARS to an IM-style transport and to expand the user base so as to be able to conduct more realistic evaluations.

References

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

Albert-László Barabási, Hawoong Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. On the topology of the scientific collaboration networks. *Physica A*, 311, 2002.

Joseph P. Bigus, Don A. Schlonagle, Jeff R. Pilgrim, W. Nathaniel

Mills III, and Yixin Diao. ABLE: A toolkit for building multiagent autonomic systems. *IBM Systems Journal*, 41(2):350–371, 2002.

Jacqueline J. Brown and Peter H. Reingen. Social ties and word-of-mouth referral behavior. *Journal of Consumer Research*, 14: 350–362, 1987.

Noshir Contractor, Dan Zink, and Mike Chan. IKNOW: A tool to assist and study the creation, maintenance, and dissolution of knowledge networks. In Toru Ishida, editor, *Community Computing and Support Systems*, pages 201–217. Springer-Verlag, Berlin, 1989.

Keith Decker, Katia Sycara, and Mike Williamson. Middle-agents for the Internet. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 578–583, 1997.

Gerhard Fischer and Jonathan Ostwald. Knowledge management: Problems, promises, realities, and challenges. *IEEE Intelligent Systems*, 16(1):60–73, 2001.

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

Michael N. Huhns, Uttam Mukhopadhyay, Larry M. Stephens, and Ronald D. Bonnell. DAI for document retrieval: The MINDS project. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 249–283. Pitman/Morgan Kaufmann, London, 1987.

Alaina Kanfer, Jim Sweet, and Anne E. Schlosser. Humanizing the net: Social navigation with a "know-who" email agent. In *Proceedings of the 3rd Conference on Human Factors and the Web*, 1997.

Henry Kautz, Bart Selman, and Al Milewski. Agent amplified communication. In *Proceedings of the National Conference on Artificial Intelligence*, pages 3–9, 1996.

Henry Kautz, Bart Selman, and Mehul Shah. The hidden Web. *AI Magazine*, 18(2):27–36, 1997.

Beom Jun Kim, Chang No Yoon, Seung Kee Han, and Hawoong Jeong. Path finding strategies in scale-free networks. *Physics Review E*, 65(027103), 2002.

Jon M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 163–170, 2000.

Bruce Krulwich and Chad Burkey. The ContactFinder: Answering bulletin board questions with referrals. In *Proceedings of the National Conference on Artificial Intelligence*, pages 10–15, 1996.

Daniel Kuokka and Larry Harada. Matchmaking for information agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 672–678, 1995.

Mark Maybury, Ray D'Amore, and David House. Automating the finding of experts. *Research Technology Management*, 43(6): 12–15, 2000.

David W. McDonald and Mark S. Ackerman. Expertise recommender: A flexible recommendation architecture. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW)*, pages 231–240, 2000.

Stanley Milgram. The small world problem. *Psychology Today*, 2: 60–67, 1967.

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, 2000.

Mark E. J. Newman. Who is the best connected scientist? a study of scientific coauthorship networks. *Physics Review E*, 64(016131), 2001.

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, 2001.

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

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

Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

Ben J. Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 158–166, 1999.

Onn Shehory. A scalable agent location mechanism. In *Proceedings of the 6th International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, pages 162–172, 1999.

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 ACM SIGCOMM*, pages 149–160, 2001.

Troy Tassier and Filippo Menczer. Emerging small-world referral networks in evolutionary labor markets. *IEEE Transactions on Evolutionary Computation*, 5(5):482–492, 2001.

Adriana Vivacqua and Henry Lieberman. Agents to assist in finding help. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 65–72, 2000.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of 22nd International Conference on Distributed Computing Systems*, pages 5–14, 2002.

Bin Yu and Munindar P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18 (4):535–549, 2002.

Bin Yu, Mahadevan Venkatraman, and Munindar P. Singh. An adaptive social network for information access: Theoretical and experimental results. *Applied Artificial Intelligence*, 17(1):21–38, 2003.