Search in Referral Networks

Bin Yu and Munindar P. Singh

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

Abstract. Referral systems have been proposed to assist people in finding potential experts in person-to-person social networks, in which each user is assigned a software agent and software agents help automate the process through a series of referrals. However, most of the existing referral systems have focused on the referral information generation, and simply consider referrals through path search in a static graph. An important question is how to efficiently search the social networks with the help of software agents, while agents only rely on their local knowledge. In this paper referral networks are proposed to model the social structure emerging among software agents. We study the referral networks empirically for the community of AI scientists (as defined in bibliographic data), and show how to control the searching process by adaptively choosing the referrals.

1 Introduction

Research on the diffusion of information indicates that interpersonal communication acts as an important channel for gathering information [5, 9]. But if we wish to rely on interpersonal communication, we still need to figure out how to determine the right person of whom to ask a question. Usually we cannot find the potential expert(s) directly, and we need some assistance from our friends or friends' friends to locate them. The phenomenon of *Six Degrees of Separation* tells us that there is a chain of only about six people between any two people in the country[17]. This relatively small value indicates that it is possible to use some intelligent software agents, who can interpret the links between people and follow only the relevant ones, to find the desired experts efficiently [2].

The importance of referrals on information flow, e.g., in marketing, has been known for a long time [5, 16, 21]. The ideas of referral systems have also been around since the early 1980s, e.g., MINDS [4, 7] and ReferralWeb [11, 12]. In a referral system each user is assigned a software agent and software agents help the people find potential experts through a series of referrals. MINDS is the earliest agent-based referral system of which we have knowledge, but ReferralWeb first explicitly used referral chains to find some people with the needed information[10]. In general, MINDS is giving much attention to learning heuristics for referral generation while ReferralWeb focuses on the problem of how to bootstrap the referral system. Neither of them studies the dynamics of social structure emerging among software agents, especially how to efficiently search the social network with the help of agents [24, 29, 31].

A referral network is simply a “loose” multiagent system, in which each agent is associated with a user and the agents can automatically generate and follow referrals upon some queries. The agents can learn on the shoulder of the user, and they can adaptively choose their neighbors or update the neighbors’ information, i.e., expertise, and sociability, upon the feedbacks from the user. Moreover, there is a good chance to find the needed experts in a referral network than in a human social network. Part of the reasons is that the query is routed among the software agents in the referral network, and the user need not worry about disturbing other users, who are shielded from seeing irrelevant messages by software agents.

Referral networks are intended to be used in a distributed, and dynamic system in which the structure of the system itself and the member of the system are capable of dynamically changing. Therefore, traditional approaches like a centralized database or matchmaker is not feasible in these settings. Similar problems were studied in other areas and each of them focused on one interesting side of the networks, e.g., peer-to-peer networks (identical responsibilities of each node) [20, 22, 25, 28], power-law networks (link distribution of each node) [1, 13], and small-world networks (shortcuts among any two nodes) [14, 15].

Peer-to-Peer Networks: Distributed searching algorithms in Gnutella [1] use a brute force searching and broadcast the request to all the peers in the network. Chord [25], CAN [20], and Pastry [22] studied distributed hashing, in which given an object, the algorithm will guarantee to locate a node (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. For all these techniques, the routing table for each node is fixed and therefore the network is not reconfigurable. In our approach, the neighbors are chosen based on how well they respond the queries from the given agent. Moreover, some networks like referral networks we studied, or more broadly the social networks, can not be partitioned by IP address.

Yang *et al.* [28] studied performance and tradeoff of three search techniques in peer-to-peer networks: iterative deepening, directed BFS, and local indices. The *directed BFS* is similar to ours, but for their approach, each node only maintains relatively simple statistics for its neighbors, i.e., the number of results that were received through the corresponding neighbor for past queries, or the latency of the connection with that neighbor. Instead, in order to select neighbors more accurately, we model each neighbor in the vector space model. Moreover, Yang *et al.* did not consider the topology of the networks, while in referral networks, software agents cooperate together to direct requests toward appropriate service or person with shortest paths if such short paths exist.

Power-Law Networks: Adamic *et al.* [1] and Kim *et al.* [13] studied the power-law link distribution of the networks, and introduced a number of local search strategies which use high degree nodes. We argue that these intuitive strategies might be helpful for people, who can decide to ask the friends who are better connected than others, but they cannot readily be used in the design of referral networks. Our approach captures the intuition via the notion of *sociability* and each agent learns about which of its neighbors is better connected.

[1] http://gnutella.wego.com/

Small-World Networks: *Small-world phenomenon* has been known for a long time, however, an explanation of the phenomenon [17] remains relatively unexplored until 1998. Watts & Strogatz [27, 26] 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). Later, Kleinberg [14, 15] found that it was possible to find short paths for the model only after randomly rewiring a two-dimensional lattice in a decentralized fashion. The topology of referral networks is similar to a two-dimensional lattice, but in our settings there is no global information about the position of the target, and hence it is not possible to determine whether a move is toward or away from the target.

The rest of this paper is organized as follows. Section 2 defines some backgrounds and notations which will be useful for the remainder of the paper. Section 3 describes some preliminary results we have got in the simulation. The future work for our research is given in Section 5, and is followed by a short conclusion.

2 Framework

In referral networks each user is represented both by expertise (ability to provide good service) and sociability (ability to provide good referrals). Queries from the user are first seen by his agent who decides the potential contacts to whom to send the query. The agent who receives a query can decide if it suits his user and let the user see that query. If not, the agent may respond with referrals to other users.

A query includes the query content as well as the requester's contact information (i.e., email address). A response may include an answer or a referral, or neither (in which case no response is needed). Note that the referring process in referral networks is different from that in peer-to-peer networks. In referral networks, all referrals are sent back to the requesting agent and the requesting agent can decide which referral should follow next and when to stop the referring process.

The initial queries by an agent are sent only to its immediate *neighbors*. It must therefore keep models for these neighbors. In addition, the agent may keep more models of other agents as well. The models of these other agents are considered the agent's *acquaintance*. The number of neighbors for each agent is limited, but the number of acquaintance is unlimited. The models for each neighbor and acquaintance are learned from interactions with the other agents, e.g., when they ask or answer a query. The acquaintances are not directly contacted for a query, but may be contacted if they are referred to by someone else. By maintaining additional models of acquaintances, the agent in effect tracks these acquaintances who may become closer associates.

2.1 User Modeling

We adapt vector space model to locate people rather than documents. In our formulation, each agent maintains two kinds of models: a *profile* for its user; and an *acquaintance model* for each of its neighbors or acquaintances. The expertise of each user is modeled as an expertise vector. Given a query vector and an expertise vector, the similarity between the two vectors is defined as the cosine of the angle between those vectors. We give a slightly different definition. This definition also captures the cosine of the angle, but scales it by the length of the expertise vector. The idea is that a person whose expertise for a query is twice as large as another person is twice as desirable as a source of information. Under the traditional model, two users would be considered equally desirable as long as the angles of their expertise vectors were the same.

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

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

For example, given 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 traditional vector space model, the two expertise vectors have equal similarity with the query vector Q, but in our framework, E_2 is better than E_1 since $Q \diamond E_2 > Q \diamond E_1$.

When a user agent receives a query, it tries to match the query against the expertise vector in its *profile*. If there is a good match, the query is passed on to its owner directly.

Definition 2. Given a query vector Q and a threshold ω_i (for filtering), where $0 \leq \omega_i \leq 1$, it says there is a good match between the user P_i and the query Q if $Q \diamond E_i \geq \omega_i$.

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 what weight is assigned to the user's sociability, which reflects how much we can trust the referrals produced by this user.

Definition 3. Given a query vector Q, the relevance of Q to any neighbor P_j of the user is computed as $Q \triangle P_j = (1 - \eta)(Q \diamond E_i) + \eta S_j$, where S_j is the sociability of the neighbor p_j; and η and $(1 - \eta)$ are the weights given to the sociability and expertise, respectively.

Further, the user may be allowed to specify an absolute relevance threshold. 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 other users. Note that usually we have$\Omega_i \geq \omega_i$.

Definition 4. Given a query vector Q and a threshold Ω_i (for referring), a neighbor P_j of user P_i is relevant to Q if $Q \triangle P_j \geq \Omega_i$ for a special value of η.

2.2 Referral Graph

The acquaintance model and profile must be learned by the agent through the interactions of its users with other users. A *referral graph* encodes how the computation spreads as a query originates from an agent and referrals or answers are sent back to this agent. Note that we will use the terms user P_i and agent A_i interchangeably, except when we are referring to users who will give an answer.

Definition 5. A referral r to agent A_j returned from agent A_i is defined 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 call $\langle A_r, A_i \rangle$ a referral, where the requesting agent A_r sends a query to one of his neighbors A_i. Let Q be a query from agent A_r. We assume that, after a series of l referrals, colleague A_j produces a response R. The entire referral chain in this case would be $\langle A_r, \ldots, A_j \rangle$, with the length l. We use the *depth* of a referral to represent the order in which agents are contacted in a referral graph (the root is the requesting agent with depth zero). Sometimes a referral may cause a cycle if it is appended to the referral graph. We can avoid cycles by representing the *ancestor* of an agent in a referral graph.

Definition 6. Given a referral chain $\langle A_r, A_{r+1}, \ldots, A_j \rangle$ from A_r to A_j, then any node A_k (except for A_j itself) on the referral chain from A_r to A_j is called an *ancestor* of agent A_j, denoted as *ancestor(A_j)*. If any agent A_k is an ancestor of agent A_j, then A_j is a descendant of agent A_k, denoted as *descendant(A_k)*.

Definition 7. A referral graph G is defined as a directed graph $G(A_r, \Lambda, R, Q)$, where A_r is the requesting agent, Λ is a finite set of agents $\{A_1, A_2, \ldots, A_n\}$, R is a set of referrals $\{r_1, r_2, \ldots, r_m\}$ and Q is the query (A_i ($1 \leq i \leq n$) is called a node in the referral graph G. Similarly, r ($\in R$) is called an edge in the referral graph G.).

Definition 8. Given a referral graph $G(A_r, \Lambda, R, Q)$, a referral $r = \langle A_i, A_j \rangle$ is *redundant* for G if and only if (1) agent A_i and A_j are some nodes in graph G, and (2) A_j is one of the ancestors of A_i.

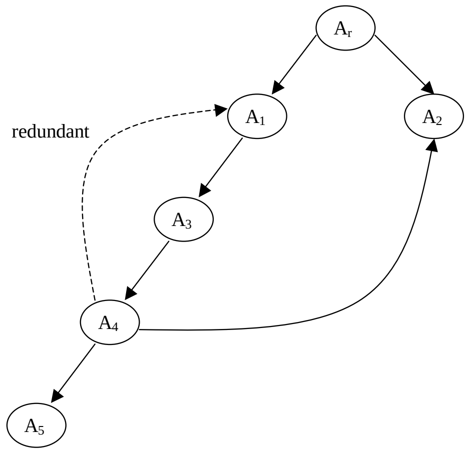

Fig. 1. The redundant referrals in a referral graph

Figure 1 shows an example of redundant referrals in a referral graph, where referral $\langle A_4, A_1 \rangle$ is redundant since A_1 is one of ancestors of A_4. Referral $\langle A_4, A_2 \rangle$ looks like a redundant referral, but it is not.

Definition 9. A given referral graph $G(A_r, \Lambda, R, Q)$ is *non-redundant* if and only if for any referral $r \in R$, r is not redundant for G.

2.3 Weighted Referral Graph

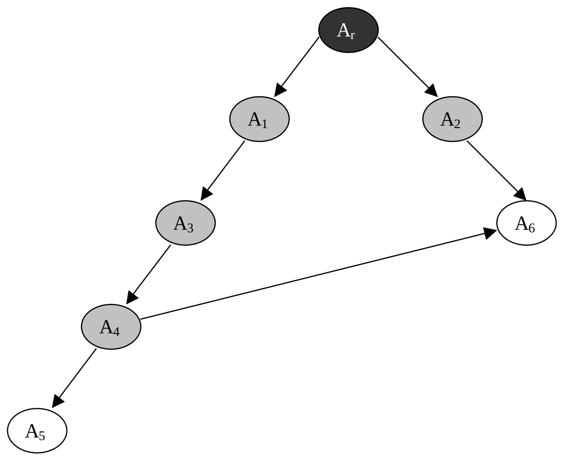

Fig. 2. The order of A_5 and A_6 in a referral graph

If we look at the referral graph more carefully, we can find all agents can be categorized as follows: (1) requesting agent A_r (black); (2) agents who have been visited (gray); (3) agents who have not been visited (white). One interesting question is for each leaf agent (who has not been visited) in the referral graph, need A_r expand all of them? Suppose each time one agent will return four referrals, there could be at most $4*4*4*4 = 256$ leaf agents for a referral graph with maximal depth four. Is there any way to control which leaf agent should be expanded first, so that the requesting agent can send fewer messages, but do not sacrifice the possibility of locating the desired experts.

Figure 2 shows a simple referral graph with only two leaf agents A_5 and A_6, the question is which one should be expanded first. One intuition is that if both A_2 and A_4 said A_6 was good, it is very likely A_6 is better than A_5, and A_r should ask A_6 first. Following this intuition, we introduce weighted referral graphs in which each node (agent) and edge (referral) are assigned a weight.

Definition 10. Given a *non-redundant* referral graph $G(A_r, \Lambda, R, Q)$ and a referral $r = \langle A_i, A_j \rangle$, then the weight of referral r is defined as $w_r = Q \triangle P_j$, which was computed at the side of agent A_i and returned to A_r.

Next we define the weight of each agent in the referral graph. The default weight of requesting agent A_r is 1.

Definition 11. Given a referral graph $G(A_r, \mathit{\Lambda}, R, Q)$, for any agent A_j, the weight of agent A_j is defined as $w_j = \sum w_i * w_r$, for all referral $r = \langle A_i, A_j \rangle$ to A_j, where w_i and w_r are the weighs of agent A_i and referral r, respectively.

Definition 12. A weighted referral graph $\bar{G}$ is defined as a non-redundant referral graph $G(A_r, \mathit{\Lambda}, R, Q)$ and plus a weight mapping $A_i | r \longrightarrow w$ for any agent A_i and referral r, where $w \in (0, \propto)$.

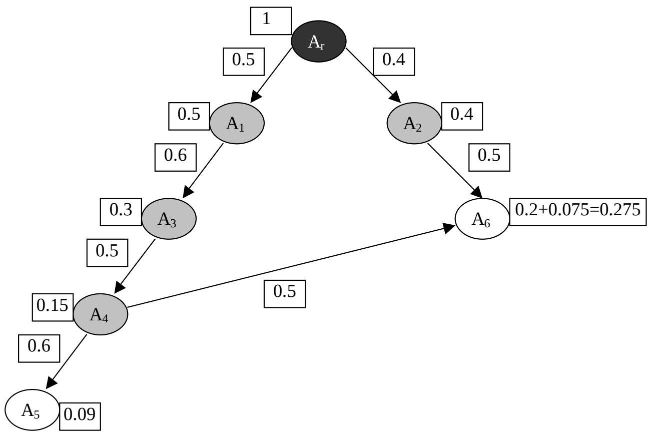

Fig. 3. Weights of each node (agent) in a weighted referral graph

Figure 3 shows an example of a weighted referral graph. The weight of agent A_6 is $0.4{*}0.5{+}0.15{*}0.5 = 0.275$, while the weight of A_5 is 0.09. We should be careful when we try to append some referrals to a referral graph. Suppose in Figure 3, $\langle A_2, A_6 \rangle$ comes back first. When we append $\langle A_4, A_6 \rangle$ to the referral graph, we have to recompute the weight of A_6. In order to facilitate the referring process, each referral graph has exactly one pool to store all non-visited nodes (leaf agents), and there could be more than one pool for the requesting agent A_r.

Definition 13. Given a weighted referral graph $\bar{G}(A_r, \mathit{\Lambda}, R, Q)$, and a new referral $r = \langle A_i, A_j \rangle$, A_j is a *cut-point* node of $\bar{G}$ if and only if $A_j \in \mathit{\Lambda}$.

In the above case, A_6 is the last agent on the referral chain $\langle A_r, A_2, A_6 \rangle$. It will be more complex if agent A_4 refers to A_2 instead of A_6, which has been visited before and returns the referral to A_6 (here A_2 is a *cut-point* node). Can we just update the weight of A_2 and then stop. The answer is NO. We have to update A_6 since A_6 is the descendant of A_2. An operation *relax* is introduced to check each descendant of the cut-point node or himself, and update their weights if needed.

```
Assumption: An agent who gives an answer won't make a referral.

Suppose agent $A_r$ is the requesting agent, set $\Lambda$ is the agents being visited. Then given a
series of referral $\{r_1, r_2, \ldots, r_m\}$, for each referral $r = \langle A_i , A_j \rangle$, agent $A_r$ will update the
expertise and sociability of other agents according to the following rules,

    (1) If $A_j \notin \Lambda$ and $A_j$ returns an answer, then
            1) appends r to the referral graph, and $A_j$ into $\Lambda$,
            2) evaluates the answer and updates the expertise of agent $A_j$ and the
               sociability of any agent on the referral chain to agent $A_j$;
            3) otherwise go to (2),
    (2) If $A_j \notin \Lambda$ and $A_j$ doesn't return an answer, then just appends r to the referral graph,
        and $A_j$ into $\Gamma$; otherwise, go to (3)
    (3) If $A_j \in \Lambda$ and $A_j \neq$ ancestor ($A_i$), then appends $r_k$ to the referral graph, and $A_j$ into
        $\Lambda$, relax $A_j$ and descendants of ($A_j$); otherwise, go to (4)
    (4) Ignore the referral r.
```

Fig. 4. An algorithm for constructing a referral graph

Definition 14. The operation *relax* (node A_i) is defined as: (1) update the weight of node A_i; (2) if node A_i is not a leaf agent, for each child of node A_i, relax(child(A_i)).

Figure 4 shows an algorithm of constructing a referral graph upon a series of referrals. Note that we actually consider the length of referral chains when we tried to expand some leaf agents in a referral graph. Our algorithm prefers the leaf agents with shorter referrals if both agents have no accumulating weights from others.

2.4 Credits/Penalties Propagation

Now we talk about how the requesting agent propagates the rewards/penalties to all relevant agents in case there are some answers returned. First we introduce an operator Π, which was used to update the sociability. The intuition behind the formula is that we want to model the change of sociability which is hard to be built up but easy to be destroyed. Also the maximal value of the sociability should be bounded by one, no matter how many successful answers come through the agent.

Definition 15. Given any two variable X and Y, where $0 \leq X \leq 1$, the operator Π was defined as $\Pi(X, Y) = X + Y - XY$ if $0 \leq Y$, or $X + XY/(1 + Y)$, otherwise.

Given a referral graph G (or $\bar{G}$), suppose user P_j returns an answer R, then the requesting agent A_r will update expertise and sociability as follows, where α is the rating given by the user, β is the learning rate, $-1 \leq \alpha \leq 1$, and $0 \leq \beta \leq 1$.

– *Expertise:* On the side of user P_r, agent A_r will update the expertise vector for P_r in the profile as $(1-\beta)E_r+\beta Q$, and the expertise vector for user P_j as $(1-\beta)E_j+\alpha\beta R$. On the side of user P_j, agent A_j will update the expertise vector for P_j as $(1-\beta)E_j+\beta R$.
– *Sociability:* Suppose l is the depth of A_j, the following algorithm will propagate the credits to his ancestors according to the distance to him, invoked by backProp $(A_j, l-1, \alpha)$.

```
backProp (Node $A_i$, int l, double credits) {
    for each parent of node $A_i$ {
        if $(l \geq 0)$ and $(i \neq A_r)$ then {
            Node $A_j$ = parent($A_i$);
            $S_j = \Pi(S_j, credits)$
            backProp(parent($A_j$), l-1, credits/2)
        }
    }
}
```

Note that we give more rewards or penalties to the agents who are near to the answering agent. For example, in figure 3, suppose A_6 is referred by A_2 first and A_6 returns an answer with quality α, then agent A_2, and A_4 will get credit α, respectively. A_3 and A_1 will not get any credit since they are too far from A_6. However, if A_6 refers to A_4 (A_4 does not refer to A_6) and A_5 returns an answer with quality α, then A_4 and A_6 will get credit α, A_3 and A_2 will get $\alpha/2$ and A_1 will get $\alpha/4$.

If there is no answer from user P_j, there will be no penalties for the expertise of agent A_j, and sociability of all members on the referral chain. In our another paper we introduce another parameter, called *cooperativeness factor*. We decrease its cooperativeness factor if there is no answer returned [30].

3 Experimental Results

The networks of scientific collaborations have been studied recently by Newman [19], and Barabási *et al.* [3]. They focused 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, and the time evolution of these qualities. In our experiments we reconstructed the social networks for all AI scientists and then evaluated the performance of expert location techniques starting with the networks.

The data is from the proceedings of AAAI(1980-2000) and IJCAI(1981-2001) conferences. In referral networks, one author is considered another author's neighbor if they have coauthored one or more papers together. However, some authors may know each other, but they may never coauthor a paper in the proceedings. In order to catch this phenomenon, we introduced some random links among the authors. In our initial networks, the number of random links equals the number of links due to coauthorship. Further, we initialize the expertise vector for each author depending on the number of papers written by the author and the total number of papers in that domain.

In our dataset, we have 4933 scientists whose papers are classified according to the 19 domains of AI[2]. For any expertise vector $E_j = \{e_1, e_2, \ldots, e_n\}$, each element e_i ($1 \leq i \leq n$) is weighted using the "TFIDF" (term-frequency inverse document frequency) [23]. That is, the weight of e_i in a vector E_j is derived by multiplying a term frequency ("TF") component by an inverse document frequency ("IDF") component. In our settings there are two cases, (1) if E_j is an expertise vector in user P_j's profile, then for any e_k, $e_k = tf_{P_j} * idf_k$, where $tf_{P_j} =$ number of papers authored by P_j in domain k, and $idf_k = log(N/n_k)$, where $N = 4933$, and n_k equal to the number of papers in domain k; (2) if E_j is an expertise vector in user P_i's acquaintance model for P_j, then for any e_k, $e_k = tf_{P_j} * idf_k$, where idf_k is similar as above, but $tf_{P_j} =$ is defined as the number of papers coauthored by P_i and P_j in domain k.

The original data is available in HTML format.[3] First, we automatically converted them into bibtex files with four fields for each entry: author, title, keyword and pages. We then categorized each paper into one of the 19 domains based on the AI ontology we constructed. For example, a paper with the keyword "case-based reasoning" will be identified as "knowledge representation and reasoning."

Next, we manually checked the consistency of the names. One assumption is that no two authors have the same names. We did five kinds of clean up: (1) spelling differences for the first name, e.g., "Dimitri Achlioptas" appeared in AAAI-01, but in AAAI-00 it was written as "Dimitris Achlioptas"; (2) first name did not always use abbreviated form, e.g., "A. Bagchi" in AAAI-88 and "Amitava Bagchi" in AAAI-86; (3) middle name did not always be used, e.g., "Ranan Banerji" in IJCAI-81 and "Ranan B. Banerji" in IJCAI-87; (4) inconsistent ordering of first and last name, e.g., "Liu Xuhua" in IJCAI-89 and "Xuhua Liu" in IJCAI-95; (5) harder cases for which we had to consult the web or ask the author(s) directly, e.g., "M. Fox" appeared in AAAI-86 should be "Mark S. Fox," instead of "Maria Fox."

Third, we identified the experts among the 4933 authors in the whole dataset. An author is an expert in one domain k if and only if the weight of e_k is above a certain threshold. In our case, the threshold was set as 8, and about 223 out of 4933 authors were identified as "experts." The rating α is equal to 1 if an expert found. The other two thresholds ω_i (for filtering) and $\mathit{\Omega}_i$ (for referring) for each agent A_i are both set as 0.1. 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, $[1, 0, \ldots, 0]$ would be a query in the domain of "AI architecture." Usually each author only has papers in one or two domains. Therefore the queries could be distinguished into two categories: *Home queries*

[2] The 19 domains and corresponding number 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), vision and robotics (660).

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

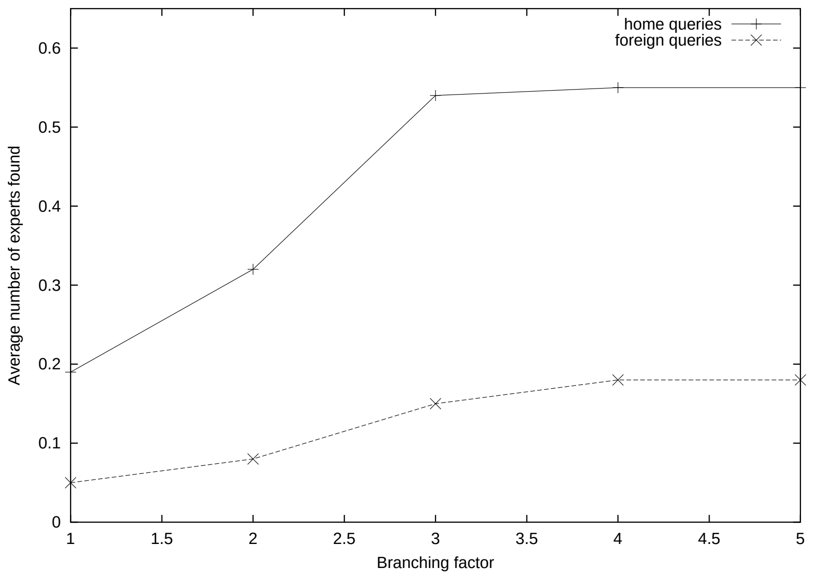

Fig. 5. Average number of experts found for different branching factors

- queries from the same domain where the author has some papers; *Foreign queries* - queries from other domains where the author has no papers.

3.1 Effect of Branching Factors

The first question is that how many neighbors will need to be referred by an agent during the referring process. Following Kautz et al., we called this the *branching factor* and denoted it by F. Figure 5 shows the number of experts found (averaged over all agents) for different branching factors when keeping the depth of referral graph as 6. We found that, for home queries, the number almost reached the highest value when $F = 3$, but for foreign queries, at least 4 was needed for F. In the following experiments the value of F (unless specified) will be 4 for both home and foreign queries.

3.2 Depth of Referral Graphs

The next natural question is how deep do the referral graphs need to be for this network? We found that for home queries, the depth of referral graphs needs to be at least five, but for foreign queries, the depth needs to expand to six. Figure 6 illustrates the power of referral chaining. For home queries, if an agent only asks his neighbors directly, he only has a 14% chance of finding an acceptable answer. But with the help of software agents in referral networks, the agent has a 55% chance, reflecting a significant improvement. Referrals are even more important when seeking an expert in an area that is different

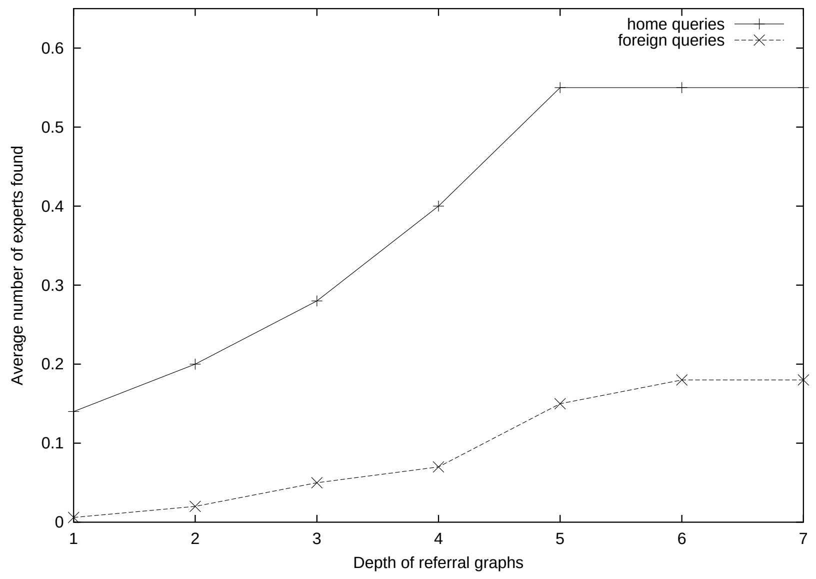

Fig. 6. Average number of experts found for different depth of referral graphs

from one's own area. For foreign queries, we find that there is an almost 30 times greater chance of success when using referrals than by just asking neighbors directly (18% versus 0.6%).

3.3 Accuracy of Referral Chains

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. Sociability was introduced to credit the ability to give good referrals. During the referring process we consider both expertise and sociability. The authors send queries, referrals, and responses to one another, all the while learning about each others' expertise and sociability. After each agent sends out about 10 home queries or 10 foreign queries, we run the experiment again for different depths of referrals. Figure 7 shows that, even with only 10 queries, the number of experts found can be surprisingly improved. This suggests learning could be effective in practice, especially for home queries. Note that the number of neighbors for each agent remains constant, but the set of neighbors is updated to promote the most promising acquaintances to be neighbors.

3.4 Minimizing Referral Graphs

Our last experiment will talk about how to minimize the referral graphs. In our third experiment we appended all non-redundant referrals to the referral graph. One interest-

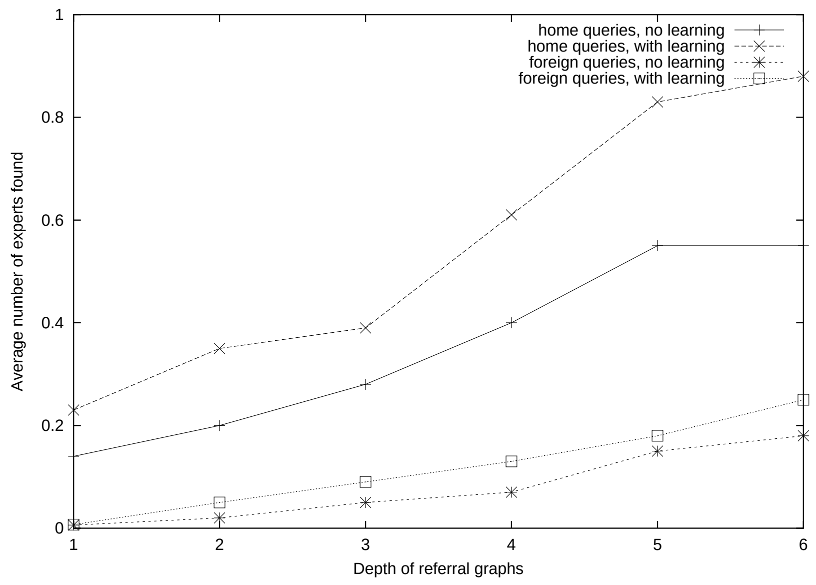

Fig. 7. Average number of experts found in a dynamic referral networks

ing question is for each leaf agent in the referral graph: need we expand all of them? In this setting we introduce weighted referral graph and the weights of each referral and agent are depended on *expertise* and *sociability* of each agent and the topology of the referral graph. When the requesting agent A_r receives a series of referrals, each time A_r will choose referrals with the highest weight one by one. The referring process stopped when one expert was found. Figure 8 summarized the results when $F = 4$. We can find that average number of referrals for per expert found is significantly improved after minimizing the referral graph.

4 Conclusion

We study the referral networks empirically and show how to control the searching process by adaptively choosing the referrals in terms of expertise and sociability. One reason to believe that referral networks would be useful is that it basically models the manner in which expertise location actually works, while allowing more people to be contacted without causing unnecessary disturbance. We discussed the upper bounds of the number of the neighbors should be selected in each step (branching factor), and the distance of search (depth of referral graphs) in search of appropriate service or person. Moreover we found the performance of the network can be surprisingly improved through learning and neighbor selection. The referral network can evolve into a larger and more effective community for information seeking through the interactions among software agents.

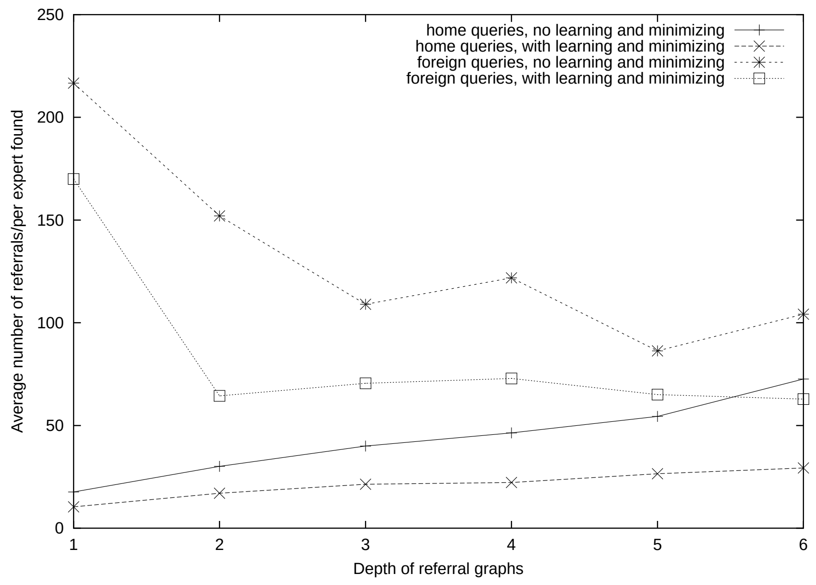

Fig. 8. Average referrals/per experts after minimizing referral graphs

The present approach to referral networks is not only useful for building social networks of humans, but we expect can also be applied in building multiagent systems in general. The conventional way to implementing a multiagent system is to use specialized agents such as brokers or facilitators [8, 6]. Despite the considerable research that has gone into the theories and architectures for such multiagent systems, there is relatively little experience with building a multiagent system with people together. A referral network approach, being perfectly decentralized, extend the stand-alone multiagent systems with users, and enable them to share their knowledge and experience on a wide scale.

We hope that our work will stimulate further studies of referral networks. Referral networks help to develop an effective, naturally occurring knowledge management system in the organizations, while each agent automatically learns one another's domains of expertise. The rapid change of organizational structure makes agent-based referral networks even more important in which workers rely on their personal social networks, rather than unstable, weakening ”organization charts” [18]. Moreover, referral networks are a natural next step in the evolution of peer-to-peer computing, in which agents can help computing networks dynamically work together, i.e., prioritizing tasks, controlling traffic flow, and searching for files or objects.

Acknowledgments

This research was supported by the National Science Foundation under grant IIS-9624425 (Career Award) and ITR-0081742. We are indebted to Henry Kautz and Bart Selman for sharing the AI bibliographic data with us.

References

1. Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Physics Review E*, 64(46135), 2001.
2. Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world wide web. *Nature*, 401:130–131, 1999.
3. Albert-Lászlió Barabási, Hawoong Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 2002. to appear.
4. Ronald Bonnell, Michael Huhns, Larry Stephens, and Uttam Mukhopadhyay. MINDS: Multiple intelligent node document servers. In *Proceedings of IEEE First International Conference on Office Automation*, pages 125–136, 1984.
5. Jacqueline J. Brown and Peter H. Reingen. Social ties and word-of-mouth referral behavior. *Journal of Consumer Research*, 14:350–362, 1987.
6. 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.
7. 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.
8. Michael N. Huhns and Munindar P. Singh. All agents are not created equal. *IEEE Internet Computing*, 2(3):94–96, June 1998. Instance of the column *Agents on the Web*.
9. Elihu Katz and Paul F. Lazarsfeld, editors. *Personal Influence: The Part Played by People in the Flow of Mass Communications*. Free Press, New York, 1955.
10. Henry Kautz, Bart Selman, and Al Milewski. Agent amplified communication. In *Proceedings of the National Conference on Artificial Intelligence*, pages 3–9, 1996.
11. Henry Kautz, Bart Selman, and Mehul Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997.
12. Henry Kautz, Bart Selman, and Mehul Shah. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.
13. 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.
14. Jon Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.
15. Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 163–170, 2000.
16. Nan Lin. Information flow, influence flow and the decision-making process. *Journalism Quarterly*, 41:33–40, 1971.
17. Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.
18. 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.
19. Mark E. J. Newman. Who is the best connected scientist? a study of scientific coauthorship networks. *Physics Review E*, 64(016131), 2001.
20. 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.

21. Peter H. Reingen and Jerome B. Kernan. Analysis of referral networks in marketing: Methods and illustration. *Journal of Marketing Research*, 23:370–378, November 1986.
22. 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.
23. Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
24. Munindar P. Singh, Bin Yu, and Mahadevan Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, 2001.
25. Ion Stoica, Robert Morris, David Karger, M. Frans Kasshoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, 2001.
26. Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton, NJ, 1999.
27. Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, June 1998.
28. Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of 22nd International Conference on Distributed Computing Systems*, 2002. to appear.
29. Bin Yu and Munindar P. Singh. Emergence of agent-based referral networks (poster). In *Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1208–1209, 2002.
30. Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 294–301, 2002.
31. Bin Yu, Mahadevan Venkatraman, and Munindar P. Singh. An adaptive social network for information access: Theoretical and experimental results. *Applied Artificial Intelligence*, 2002. to appear.