A Dynamically Formed Hierarchical Agent Organization for a Distributed Content Sharing System

Haizheng Zhang Victor Lesser
Department of Computer Science
140 Governors Drive
Amherst, MA, 01003
{hzhang,lesser}@cs.umass.edu

ABSTRACT

The organization and mechanisms of agent societies are becoming increasingly important with the growing size of agent networks. Particularly, in multi-agent based content sharing system, a flat, peer-to-peer(P2P) agent organization is not the most effect agents from efficiently locating relevant agents for queries. This paper develops and analyzes a hierarchical agent group formation protocol to build a hybrid organization for large-scale content sharing system as well as a content-aware distributed search algorithm to take advantage of such an organization. During the organization formation process, the agents manage their agent-view structures to form a hierarchical topology in an incremental fashion. The algorithm aims to place those agents with similar content in the same group. We evaluate the system performance based on TREC VLC 921 datasets. The results of the experiment demonstrate a significant increase in the cumulative recall ratio(CRR) measure compared to the flat agent organization and structure.

Categories and Subject Descriptors

I.2.11 [**Aritificial Intelligence**]: Distributed Artificial IntelligenceMultiagent Systems

General Terms

Algorithms,Design,Experimentation

Keywords

Peer to Peer Networks, Agent Organization, Distributed Information Retrieval

1. INTRODUCTION

This paper investigates the role of an agent organization in a large-scale content retrieval system. In such a system, each agent shares its document collection and cooperates with other agents to conduct information retrieval tasks. An information retrieval task is defined as a process during which an agent receives a query from a user, forwards the query to other agents, who then conduct local search on their own document collections and return relevant documents to the query initiator. In this paper, we consider such a system as a multi-agent system and focus on the organizational perspectives, i.e. how to organize these agents together to provide better system performance.

Our previous results showed that an uninformative search strategy performs badly on flat P2P agent networks. However, a topology reorganization process combined with context-aware search algorithms can improve the information retrieval performance considerably [9]. This motivates us to investigate more complicated multi-agent organizations. Here we propose a hierarchical agent organization formation protocol to explicitly form a multi-level topical hierarchical structure to facilitate locating relevant documents. In one such system, the agents join different groups largely by their content similarity. Even among the same group, Agents can be placed in different levels to reflect their different collection similarity.

We make the following assumptions in this paper. First, each agent maintains an independent index and an IR search engine for its local document collection. However, we do not introduce any further restrictions on the local search engines and thus the network can be populated by agents having very different local search engines. Second, the experimental results presented are based on local search engines that are "perfect" in that they return all relevant documents in the collection for a given query. Third, we assume there is a third-party protocol in place to merge the returned results. Thus, our protocol does not have to deal with the merging of the returned lists. Lastly, we assume that agents are cooperative in that they all agree to use the same protocols for propagating resource descriptions among each other, accepting queries from peers and finally returning search results to the originators of the queries

The main contributions of this paper are as follows: (1) A group formation protocol for forming a hierarchical topical organization. The group formation is achieved by organizing the agent-view structures properly so as to place semantically similar agents together to form explicit groups in an incremental and distributed manner. (2) A content-aware search algorithm taking full advantage of the hierarchical organization. During the search process, agents in the net-

work follow various cooperation strategies to forward queries and return results in the network.

The remainder of the paper is structured as follows: Section 2 presents agent's internal structure and system architecture. Section 3 describes the algorithm to form the hierarchical groups in content sharing systems. Section 4 illustrates the content-aware distributed search algorithms. Section 5 gives the experimental settings and results analysis is given in Section 6. Section 7 discusses some design issues and future work. Section 8 presents the related work. Section 9 concludes the paper.

2. THE SYSTEM ARCHITECTURE

In the proposed hierarchical agent society, agents have two roles: group-mediator and query-processor. All non-leaf agents in the organization take on both roles while leaf agents only take on the role of query-processor. Each mediator manages a group of agents and takes on a central role in group management including decisions on whether a new agent should be added to the group, when to reorganize the group, the selection of group members to handle a query and the propagation of queries to non-group members.

Fig. 1 illustrates the internal structure of an agent. Each agent is composed of four components: the collection information, a local search engine, an agent-view structure and a control unit. The collection information includes the collection hosted by the agent to share with other agents as well as a collection model built for the collection. The collection model can be considered as the "signature" of a collection; a collection model is a statistical language model built for a particular collection. It characterizes the distribution of the vocabulary in the collection and estimates the probability of absent words using various smoothing techniques. The language model concept was originally introduced in information retrieval research [7] and has proven effective in the distributed IR applications [9, 4, 1].

The language model has many interesting properties which are easily exploitable in peer-to-peer network systems: first, a collection model is lightweight as it significantly condenses the description of the content of the collection and thus is much smaller in size compared to the collection. Additionally, the size of the collection model grows minimally with the size of the document collection. Secondly, the collection model is a relatively accurate indicator of the content of the collection. In our agent organization, collection models are propagated in the network to broaden the scope of an agent's awareness of what other information is available in the network. The agent control unit's role is to accept user queries and to decide whether the queries should be processed by one or more group members and determine the order of the other agents or groups that the queries should be forwarded to. The local search engine allows each agent to conduct a local search on its document collection so as to determine whether there are any documents that meet the criteria of a specific user query and then return relevant documents.

The agent-view structure, also called the local view of each agent, contains information about the existence and structure of other agents in the network and thus defines the underlying topology of the agent society. The agent-view

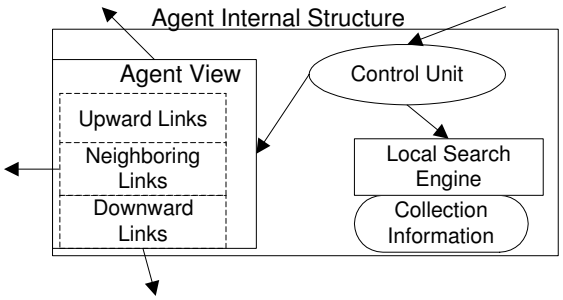

Figure 1: The internal structure of an agent

structure connects the agents together to form a specific topology in the network. Agents need to hold enough information about other agents to direct queries and yet not to introduce inconsistency between their agent-view structures and the actual changing agent information. In practice, the agent-view structure contains the collection model of the collections as well as the address of these agents. The agent-view structure also differentiates agents that take on both roles from those that take on only the role of query-processor. The agents who only take on the role of query-processor have only two kinds of links in their agent-view: neighboring links which connect neighboring agents together and upward links, which connect a member to its mediators. Besides these two kinds of links, mediators have downward links which connect a mediator to its immediate group members. In our current model, besides their direct mediators, agents also connect to the top-level mediator of their group with an upward link so as to facilitate a new agent joining and the associated group formation process, as well as the search process that will be introduced in later sections. This top-level connection is not an absolute necessity since there are many alternative linking structures that can achieve this goal. For simplicity in describing the protocol, we assume the existence of this type of link. The hierarchical group formation process organizes the agent-view structures of the agents to form a nearly-decomposable hierarchical structure.

It is worth to pointing out that the degree, or the number of the entries in agent-view structures is an important factor in determining system performance. On one extreme, if each agent has the information about all agents in the network, it is indeed a centralized version of distributed information retrieval problem which has been well studied in the information retrieval community[1]. This, however, is impracticable in P2P system for a very large network. Previous studies indicate that in general the degree of each node satisfies the power law distribution. In our work, we specify that each agent has a degree limit for neighboring links based on previous work. Further we assume there is underlying linear relation between the neighboring links degree and upward, downward links degrees. In this paper, we use a discount factor to capture this correlation. In the experimental setting and results analysis, we will compare the performance for different discount factors settings.

Figure 2 illustrates an agent organization with three levels.

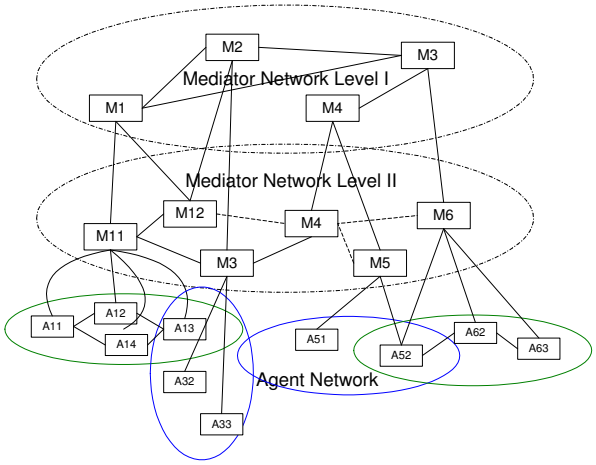

Figure 2: The system architecture

In the hierarchical structure, we specify that each group is a cluster of the agents with similar topics. Each agent can belong to multiple groups if applicable.

In our nearly-decomposable hierarchical organization, the members at the same level are connected to each other in a peer-to-peer fashion despite the fact that agents are connected in a hierarchical fashion among different levels. These hybrid links make the system well connected, more load-balanced, and meanwhile have clusters of agents that are organized by topical structures.

3. HIERARCHICAL GROUP FORMATION

In this section, we propose an online hierarchical grouping protocol. The protocol aims to build a topical hierarchical structure incrementally as agents join the network. For a particular new agent, the joining process involves locating an appropriate group (or starting a new cluster) and the actual joining process. During this process, the new agent builds its agent-view structure to place itself appropriately in the network. Meanwhile, the current existing members might prune their agent-view structure to reflect the changes and connect themselves to the new agent through neighboring links. This section is organized into three subsections. Section 3.1 introduces the message propagation and group locating process; Section 3.2 describes the actual join procedure; Section 3.3 gives several examples to clarify the algorithms described in Section 3.1 and 3.2.

3.1 Message propagation and group locating

The group locating process propagates *Join?* messages in the network and returns

When a new agent joins the system, it first contacts any agent in the system with a *Join?* message. (Note that our protocol does not specify how to acquire the entry point information of the network. However, we assume the network is accessible either by an out-of-band approach as Gnutella does or by other mechanisms).

groups invitations from which the new agent chooses one or more of these invitations to join a sepcific part of the agent society. This process is carried out in a top-down fashion. Specifically, the new agent always tries to locate the appropriate top-level mediators whose similarity with the new agent is above their group thresholds. The top-level mediators will then either add the new agent as its own direct lower level member, or add it to a lower level group by passing the new agent to other members in the group. The rest of this section details the group locating process.

The *Join?* message includes the collection model and other information about the new agent. Upon receiving a *Join?* message from a new agent, say A, an agent forwards the message to its top-level mediator, say M, through the upward link described in the previous section. M compares the similarity of agent A's collection model and its group model, which is approximated by M's collection model, i.e $P(A|M)$. If the similarity is above the threshold associated with M, M will start a procedure to generate an invitation for agent A to join the group it manages.

There are two cases in this action. In the first case, if the number of the downward links for the top-level mediators is below a pre-specified limit, the top-level mediator will simply invite the new agent to join its group. In the second case, if the downward degree of the mediator has reached the pre-specified limit, the mediator then starts to merge two group members into one subgroup in order to integer the new agent. This process works as follows: the mediator picks the two semantically closest direct members A and B in its group (the new agent could be picked) and start to merge A and B. The top-level mediator sends out a *MergeRequest* message to agent B and expects either *Merged!* or *Failed!* message back from B. When B receives the merge request from the mediator, it then checks with A to join group A. Note that now member A is in the same situation as the mediator was. A then takes the same actions as the mediator took in order to add agent B in its group. If the number of the downward links of A has reached the limit or for whatever other reasons that A might not be able to add the group B, A simply returns *NotAccept* message to B. B will then report either *Merged!* or *Failed!* messages back to the mediator depending on the outcome of the group merge process.

Upon receiving the *Merged!* message, the mediator removes B from its downward links and add the new agent. Otherwise, if B failed to join A, the mediators would continue to pick another two agents whose similarity is only second to the similarity between the two agents the mediator picked and continue the same procedure. A *GroupInvitation!* message will be sent back to the new agent if the mediator finally can add the new agent in. Otherwise, a *NotAccept* message will be sent back to the new agent. Note that, during this process, if the similarity of the new agent to some of the group members is close enough, it can be picked to join the lower level mediators.

Figure 3 depicts the group locating process. The degree limit of the mediator in the example depicted in Fig. 3 is four. The current existing group members of the mediator are A, B, C, D respectively. We assume that agent A and agent B are the most similar agent pair in the group.

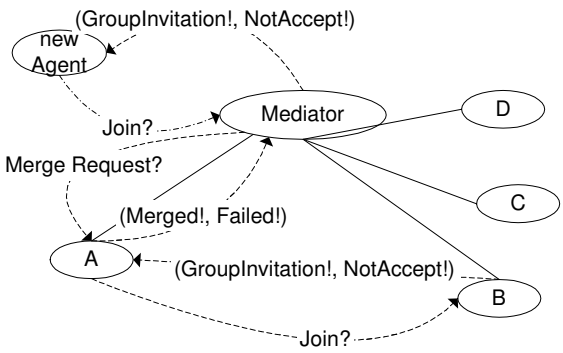

Figure 3: Group locating process

The dotted lines show the message flows during the group locating process.

3.2 Member Joining and mediator re-election

After sending out the *Join?* message, the new agent starts a timer. When the timer expires, the new agent examines the group invitations received from current agents in the network and decides which group(s) to join.

If the process ends up with no group invitations, the new agent will start a new group with a threshold $P_e(D|G)$ which is pre-acquired through offline computation. It then becomes a top-level mediator in the network. If it receives multiple group invitations, we specify that the new agent will join all the groups as long as the number of the upward links is below the upward links degree limit. At this point, the new agent would always add the top-level mediator with an upward link.

Note that we set the mediator as the content centroid for the group. With more and more nodes joining in, the content centroid changes over time. For this reason, the current mediator periodically checks its group members and determines if it should hand over the mediator position to a new mediator. If this happens, the mediator will pass to the new mediator its mediator neighbors and all the information about the regular agents in the group to the new mediator. The structure of the group might change correspondingly as the degree of the old mediator and the new mediators could be different. This update will be sent to the affected agents to update their agent-view structures to reflect the recent changes.

Once the new agent and its group mediators update their agent-view structure, the new agent is now part of the agent organization. The new agent will then broadcast *Arrival* messages in order to build intra-level connections to other agents in the group. This process results in building neighboring links for both new agents and the currently existing agents. These intra-level links can promote the connectivity of the system and thereby improving the robustness of the network.

Specifically, the new agent sends out *NewArrival!* messages with a certain TTL(time to live) value to the mediators who

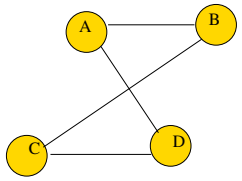

Figure 4: Before *E* joins [Scenario 1]

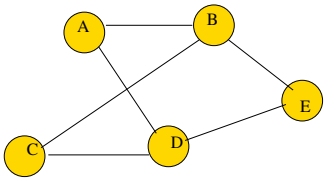

Figure 5: After *E* joins [Scenario 1]

then forward the message to their neighbors and or upper level mediators. TTL value decreases after each hop. During the message propagation process, each existing agent in the network also takes this opportunity to prune its neighboring list. Upon receiving the *NewArrival!* message, a current agent chooses to build a connection with the new agent and send out a *LinkInvitation* message with a certain likelihood. Once it decides to build a link to the new agent, the current agent simply adds the new agent as a neighbor if the current degree is below the capacity limit and otherwise, they will cancel the current links to take the new agent. Upon receiving the *LinkInvitation* message, the new agent chooses which agent to connect to randomly.

3.3 An example for group formation algorithm

Section 3.1 and 3.2 elaborate the group formation process. In this section, we give some concrete examples to further clarify the algorithms introduced in the above sections.

Scenario 1: Agent E joins in the system as a top-level mediator:

Fig. 4 and 5 demonstrate how a new agent, E, joins the agent network as a top-level mediator. In both figures, A, B, C, D are all top-level mediators. In Fig. 4, when a new agent E joins the network, it contacts any agent in the network through an out-of-band approach. If the agent is not a top-level mediator, it then forwards E's *join?* message to its corresponding top-level mediator. Particularly, assume that in Fig. 4, B is the first top-level mediator that receives join message from E. B then forwards the *join?* message among the top-level mediator network. In this example, we assume that the similarity between any top-level mediator and E is less than the thresholds associated with the top-level mediators, This assumption ends up with no any invitation message for E. Under this situation, E would start a new group with the threshold associated with the data source where E comes from. In the meantime, the other top-level mediators build neighboring connections to the new top-level mediator, E. The resultant situation after E joins the network is shown in Fig. 5.

Scenario 2: Agent F joins the group led by E

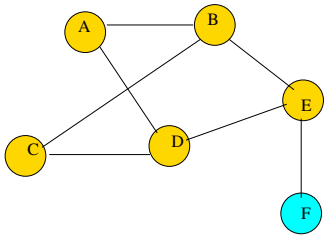

Figure 6: After F joins the system [Scenario 2]

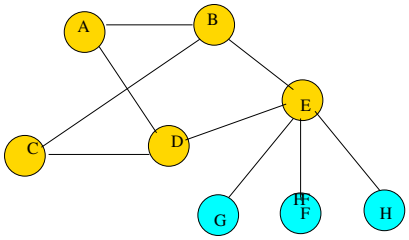

Figure 7: Before agent I joins [Scenario 3]

In this scenario, a new agent, F, joins the network shown in Fig.5. In this particular example, we assume that the similarity of the top-level mediator E with F is above the threshold associated with E. Thus F's joining process differs from E's only in that F receives invitition from E to join its group. In this case, since E does not have any direct group member yet, it simply adds F directly to its group. Fig. 6 depicts the resulted network after F joins.

Scenario 3: When the top-level mediator is fully-loaded

Fig. 7 and 8 show the situations before and after a new agent I joins the network. In this scenario, we assume that agent I repeats the same process as agent F experiences in scenario 2. However, the difference is that the top-level mediator E can not take more direct group members since the number of the direct members of E has reached the limit specified by the downward degree. Therefore, when E receives the join message from agent I, because the similarity is close enough, E identifies I as a potential member in its group. In this situation, E selects the most similar agent pair in its current group, G and H in this case, to merge. More specifically, E asks agent H to join the group led by G. Notice that this is exactly the same situation in scenario 2. After E receives the merging confirmation message from H, it then takes agent I as a new group member and removes the old downward link with H. Fig. 7 shows the situation before agent I joins the network while Fig. 8 exhibits the resulted situation that agent I joins the network.

Scenario 4: A recursive joining example

Fig. 9 and 10 illustrate another agent joining scenario to further clarify the group formation algorithm. Fig. 9 shows the situation before agent L joins the network. In this figure, both E and H can take no more direct group members. Therefore, upon receiving the join message from agent L, mediator E selects the agent pair of the highest similarity in its group to merge. In this case, we assume they are F

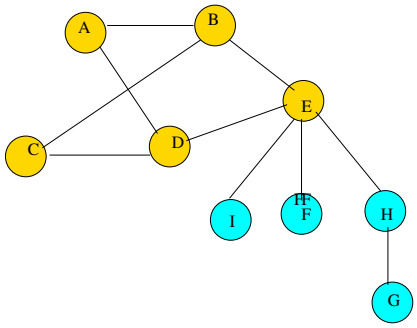

Figure 8: After agent I joins [Scenario 3]

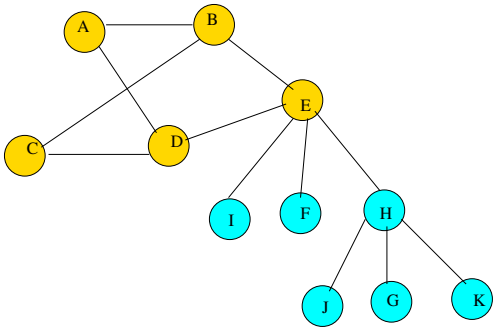

Figure 9: Before agent L joins [Scenario 4]

and H. During the merge process, F sends a join message to the group led by H. Notice that the situation of H exactly resembles that of agent E in Scenario 3. After F successfully joins H, it sends back confirmation to agent E, which then takes L as a direct group member, as shown in Fig. 10 .This recursive process can be more complicated in networks with more levels.

4. A TWO-STAGE SEARCH ALGORITHM

In this section, we first describe a two-stage search algorithm in Section 4.1 and then gives more examples in Section 4.2.

4.1 Algorithm description

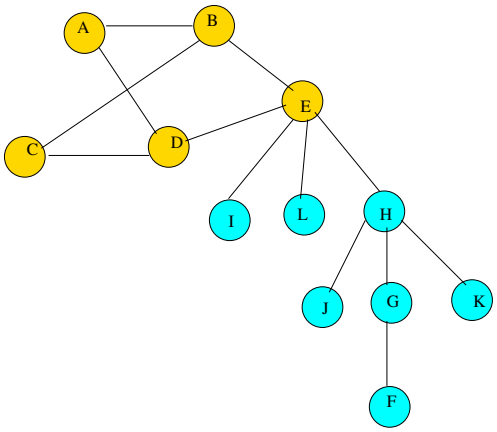

Figure 10: After agent L joins [Scenario 4]

Search in a content sharing multi-agent system should avoid significant communication costs and minimize the access to those agents that do not contain relevant documents for a given query. The benefits of communication cost savings are obvious. However, avoiding unnecessary query processing that involves local searches of documents collection can also significantly improve overall system performance especially in situations where there are many active queries concurrently being processed by the system.

Our query search process is structured into two stages. In the first stage, a coordinated search protocol is used to relocate the queries to "relevant agent zones" by taking advantage of the hierarchical structure that has been built. Here, a "relevant agent zone", as opposed to an "irrelevant agent zone", means a group whose members contain relevant documents. This is a somewhat fuzzy definition as there is no clear boundary between a relevant-agent-zone and an irrelevant-agent-zone. However, as shown in [9], in reality, most of the agents in the network are irrelevant to a given query. Therefore, in our coordinated search algorithm, we try to locate the "relevant agent zones" by sorting the $P(Q|G)$ value, which can be considered as the similarity of a query Q and a group collection model G. This value in our paper is again estimated by the similarity between the query and the mediator collection model. Specifically, the query initiator sends out the query to the top level mediators with a certain TTL(time to live) value. TTL value decreases by 1 when the message goes through each agent in the agent society. After it expires an agent will not forward the query any further, thus stopping further search along this path. Upon receiving the queries, a mediator will return with the content similarity of the group and the query $P(Q|G)$. The query initiator then picks the N highest similar top-level mediators as the starting points of the second phase search. Notice that a sorting strategy is used to pick the starting points instead of a threshold strategy used in [9] as the calculation for content similarity value between a short query and a collection[5] can be biased by the particular collection properties and thereby making an absolute threshold value inaccurate.

During the second stage, we start the search from the mediators chosen in the first phase. The query initiator forwards the query to the K most promising agents who then proceed to forward the query on to their neighbors in a decreasing order of similarity values. This process continues in the network until all the agents receiving the query drop the message or there are no other agents to forward to. There is no explicit recognition by individual agents that the query is no longer being processed by any agent in the network. The K value affects number of agents that could be visited when the search algorithm ends. The bigger K value is, the more messages will be generated in the system and potentially more agents will eventually be visited. In our experiments, we set this value as the minimum value of 8 and the number of the top-level mediators.

4.2 An example for searching algorithm

Section 4.1 presents the two-stage search algorithm. Two examples are given in order to make the algorithm more concrete.

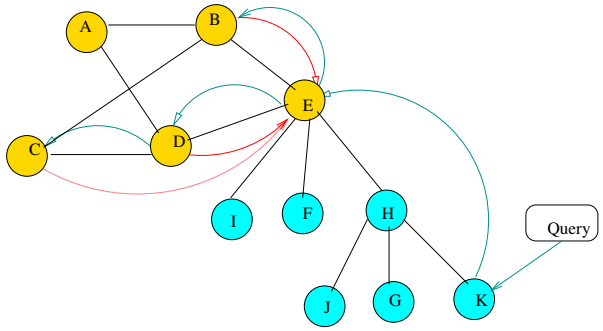

Figure 11: The first phase

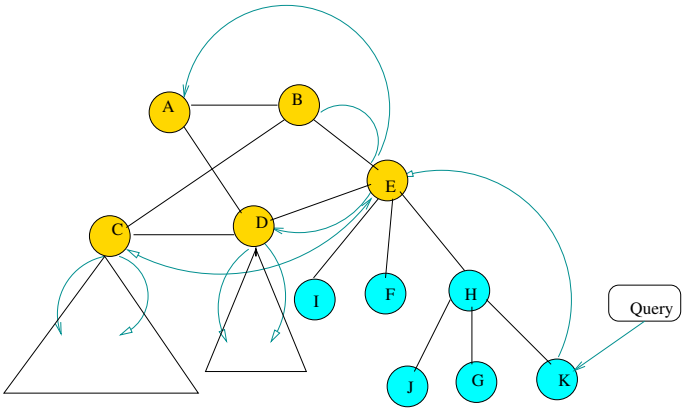

Figure 12: The second phase

Fig. 11 and 12 demonstrate the situation when agent K receives a query from its user. Fig. 11 shows the first phase of the search process described in Section 4. After receiving the query, agent K forwards it to the top-level mediator, E, connected through an upward link. Upon receiving the query, agent E forwards it out to other top-level mediators, which calculate the similarity values of their collection models to the given query and return the values to agent E.

During the second phase, the top-level mediator E ranks the similarity value it collected from other mediators. It then propagates the query to the mediators according to the ranking. The mediators then proceed to forward the query to its group members. This process is shown in Fig. 12.

5. EXPERIMENT SETUP AND THRESHOLD CALCULATION

Definition 1: Cumulative Recall Ratio (CRR) for a query after n agents are searched is defined as

$$CRR_{q_i,n} = \sum_{j=1}^{n} \frac{r_j}{R_{q_i}}$$

Here R_{q_i} is defined as the total number of relevant documents located in the entire network for the query q_i, and r_j is the number of relevant documents located at agent j. CRR is used as a metric to measure the performance of a distributed search algorithm in relationship to the number of agents the algorithm covers.

Table 1: Collection Statistics

collection size	source	Docs			Megabytes		
		Min	Avg	Max	Min	Avg	Max
921	TREC VLC	12	8157	31703	1	23	31

5.1 Experiment setup

In our experiment, we use TREC-VLC-921 dataset which contains 921 sub-collections split from TREC VLC1 collection by information sources. TREC VLC1 is part of the TREC collections which are distributed by the National Institute of Standards and Technology (NIST) for testing and comparing the current text retrieval techniques. TREC VLC1 (very large collection) includes documents from 18 different data sources, such as news, patents, and the Web[3]. We ran the query set 301-350 on TREC-VLC-921 to simulate the user queries. Table 1 shows the collection statistics for TREC-VLC-921. The distribution of relevant documents for the queries is illustrated in [9].

We use the algorithm introduced in [6] to calculate the degree limit of the agents with parameters $\alpha = 0.5$ and $\beta = 0.6$. In our experiments, we estimate the upward limit and downward degree limit using linear discount factors 0.5, 0.8 and 1.0. The corresponding performance will be presented in Section 6.

Content similarity measures are heavily used in both group formation and the distributed search algorithms. In our framework, both collection models and query models are treated as language models, and therefore, distributions. We use Kullback-Leibler (KL) divergence to measure the distance between collection models or collection models and query models. We use Lemur toolkit [5] to calculate the distance. Theoretically, KL divergence is always positive and ranges from 0 to infinity. However, the approximation approach used in [5] can end up in negative results. We use a conversion formula $W_{KL}(p,q) = 10^{-\beta D(p||q)}$ to transform the dissimilarity measure into the similarity measure [9]. Here an empirical parameter β maps the original distance value to domain $(0,1)$. After testing different values, we set β value as 10.

5.2 Threshold calculation

To determine if an agent collection model C belong to a certain group collection model G, we calculate the probability $P(G|C)$ with Bayes formula:

$$P(G|C) = \frac{P(C|G)P(G)}{P(C)}$$

Note that G represents the group collection model which is the sum of all the agent collection models in its group. In computing the above formula, as we do not have the prior probability information about $P(C)$ and $P(G)$, we simply assume the prior probability of an agent and the group is uniformly distributed. Therefore, we use probability $P(C|G)$ to estimate $P(G|C)$. In order to further reduce the computation complexity, we estimate the group collection model G with the group mediator collection model M, as it is specified as the centroid of the group, i.e. $P(C|M)$. Theoretically, this probability should be computed with appropriate smoothing techniques described in [7]. However, the computational cost prevents us from doing so in an online manner in large P2P applications. Here we estimate $P(C|M)$ with the content similarity computation which is introduced in the experimental setting section in order to reduce its computational cost.

Particularly, the top-level mediators are associated with an entry threshold $P_e(C|M)$ which is acquired through an offline computation by the minimum similarity value of M with the members from the same source.

$$P_e(C|M) = min_{c \in G} P(C|M)$$

This value is group dependent and specified prior to the group formation process is initiated. In order to join a group led by a certain top-level mediator (directly or indirectly), the probability of a new agent collection C belonging to the group G, $P_e(C|M)$, has to be above the threshold associated with the top-level mediator.

Based on the method described in Section 3 and similarity value calculation described in the above paragraph, we calculate the thresholds for 18 data sources of TREC VLC1 dataset. The thresholds are listed in Table 2.

Table 2: Thresholds for Datasets

Data Source	Thresholds
AAG(Australian Attorney-General's Department)	0.897
ADIR(Australian Department of Industrial Relations).	0.887
AFPL(Australian Federal Parliament)	0.952
AU(Websites operated by ten Australian Universities)	0.93
CRE(Congressional Record I)	0.86
CRH(Congressional Record II)	0.95
DO(DOE publications)	0.90
AP(Associated Press)	0.969
PA(U.S patents)	0.824
FBIS(Foreign Broadcast Information Service)	0.936
FR(Federal Register)	0.935
LA(LA Times)	0.974
GU(Gutenberg collection of out-of-copyright books)	0.88
SJM(San Jose Mercury News)	0.972
FT(Finicial Times)	0.968
GH(Glasgow Herald)	0.97
NEWS(USENET NEWS)	0.924
ZIFF(Ziff-Davis)	0.948
WSJ(Wall Street Journal)	0.973
WE(Downloads of websites)	0.94

6. RESULTS ANALYSIS AND EVALUATION

In experiments, we build the agent organization incrementally as the 921 agents join the system. Then we ran the search algorithms with the query set $301 - 350$. The query always starts from a randomly picked agent. We define a time unit as the communition time cost from one agent to the next one. Notice that this definition does not include

the local search time, which will be analyzed separately in the next section. During the search process, we record the visited time for each agent and the corresponding generated messages. With this data, we are able to calculate the CRR value for a certain time period. To get a greater than 95 percent confidence interval, we ran the simulation 50 rounds. In each round, we built a new agent organization and repeated every query 50 times on that organization.

The experimental results show that the semantically close agents are consistently grouped together. In this section, we analyze the resulting hierarchical group organization from both information retrieval and system performance perspectives. Note that Figure 13, 14, 15 demonstrate the expectation value from those simulation results with TTL value set as 4 in the first relocating stage in our search algorithm and 8 as the number of starting points for the second phase search.

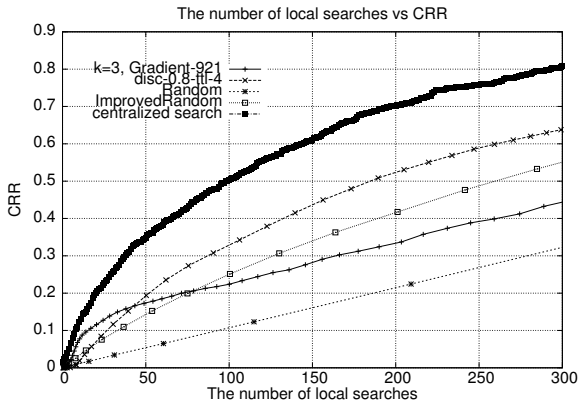

Figure 13: CRR versus the number of local searches

6.1 Performance Analysis

Recall and precision are two important measures in traditional information retrieval research. Traditional recall ratio reflects the proportion of the relevant documents over the retrieved documents in a collection while the CRR value characterizes the relevant documents returned when a certain number of agents have been searched. Therefore, the CRR metric in our system can be considered as the counterpart of recall ratio in traditional information retrieval field. As we assume that each agent is able to distinguish relevant documents from irrelevant ones with 100% precision, the traditional precision ratio corresponds to the capability of the agent selection(database selection) in P2P systems. In another word, in order to maximize the cumulative recall ratio, we need to forward the query to those agents hosting most relevant documents. In this section, Fig. 13 demonstrates the ratio of CRR over the agents retrieved. This is achieved by plotting with the pair (CRR, site searched). This measure has theoretical meaning in terms of information retrieval performance as it characterizes the collection selection performance. By searching fewer agents, we are able to save tremendous messages cost and local computational costs.

In addition to the two IR metrics, the number of messages generated and searching time t are two important measures from the system performance perspective. To simplify evaluation, we do not consider the time spent for relevant documents to return to the query initiator. Therefore, we define the the searching time t as the sum of the time spent in the trip from the query initiator to agents, i.e. t_1, and local processing time spent in agents, i.e, t_2. As the local processing involves searching the local document collections, sorting the similarity of neighboring agents to queries and forwarding queries to other agents, we believe that t_2 is the dominant factor in searching time t. Fig. 14 illustrates the number of messages generated versus CRR value. Fig. 15 depicts cumulative recall ratio versus the elapsed time in the search.

Note that in any case, we are not interested in the exhaustive searches with 100% recall ratio. Instead, we only evaluate the system when a relative small part of the system is searched. This is because that there can be tremendous relevant information sources in P2P information retrieval as Web information retrieval and thereby making an exhaustive search unnecessary. Another downside about exhaustive searches is it could lead to a bad performance in the presence of many concurrent queries in the system as it consumes more resources.

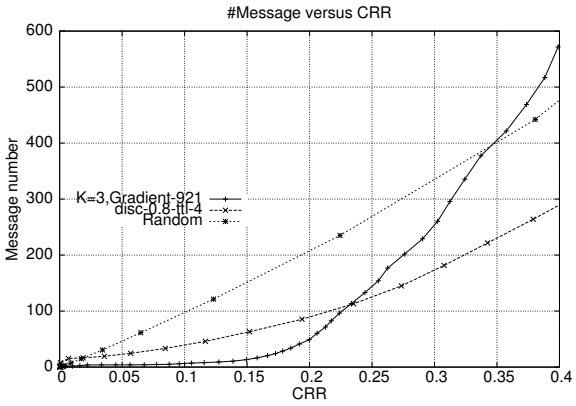

Figure 14: Messages number versus CRR

In Figure 13, we compare the results with the kNN, AVRA algorithm (K=3, Gradient Search Algorithm) which we described in our previous work[9], an improved random approach with a heuristic that agents forward queries to the neighboring agents in a decreasing similarity order(as *ImprovedRandom* in the figure), and the random approach. As it turned out that the degrees of downward and upward links did not contribute much to the system performance, we only keep the curve when the discount factor is 0.8. In Fig.13, The upper line is the central approach: Centralized KL divergence based approach is used as an upper bound on performance, which is common in the distributed IR literature. This approach assumes the network is a fully-interconnected graph and agents are visited in the order of decreasing similarity values between collections and each query. We have several observations from Fig.13: (1) The central KL approach consistently outperforms the other three approaches. The results are consistent with the conclusion that the collection model is a stable indicator for the collection from distributed information retrieval experiments; (2) As little is known about the content distribution, the number of rel-

evant documents retrieved by random algorithm is proportional to the number of agents that have been searched; (3) The graph demonstrates that the search algorithm on the hierarchical topology consistently outperforms the other approaches including AVRA, kNN and improved random approaches;(4) Notice that the sorting strategy is more advantageous compared to the heuristic we used in the previous work[9]. *ImprovedRandom* outperforms the similar strategy, i.e *kNN*, which we used in [9]. Again, as we mentioned in the previous sections, we believe the edge of the sorting strategy over threshold strategy comes from the fact that absolute content similarity is less accurate in heteregeous environments.

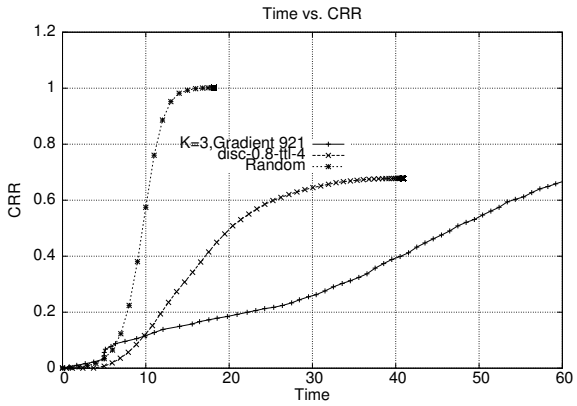

Figure 15: CRR versus communication time

Fig.14 illustrates that our approach generates fewer messages than random approach. Particularly when CRR reaches 40%, there are about 500 messages generated in the system for the random search while the two-stage search algorithm generates from $260-300$ messages depending on the parameters. An interesting fact is that AVRA algorithm performs really well when CRR is below a certain value, but increases rapidly thereafter. This benefit comes from the fact that AVRA algorithm attempts to locate a good start point for a given query with minimal communication efforts. Therefore, messages number increases slowly at the begining. However, once the query is forwarded out of relevant-agent-zone, AVRA algorithm consumes more messages in order to gain a certain amount of CRR.

Fig.15 demonstrates the time units t_1 spent to reach a certain CRR value. Not surprisingly, with the presence of only a single query, the two-stage search algorithm and AVRA algorithm spend more time in communication. This fact is attributed to the facts that (1) the two-stage search is a kind of focused search approach which spans less than the random search algorithm; (2) In AVRA algorithm, as agents of similar collection models are normally clustered together, a query tends to be forwarded to same agent, thereby makeing the query harder to reach more agents. However, we believe that in real system, both AVRA and the two-stage search algorithm would perform much better. The confidence comes from two reasons: (1) as we mentioned in the previous section, we believe that the t_1 is a minor factor contributing to the search time t considering the agent selection efficiency. (2) Fig.15 only shows the situation with a single query in the system. With many queries concurrently in the system, considering the messages queuing time, the situation in Fig 15 may not be an accurate predictor of performance. We leave this simulation as future work.

7. DISCUSSIONS AND FUTURE WORK

This work explores techniques for the dynamic creation of an agent organization and its impact on search efficiency for a distributed information retrieval applications. In this paper, we assume the agents are cooperative and therefore we are not concerned about the incentives of the agents to taking various roles in the protocol. Additionally, we did not consider several important technical issues such as load balancing and other design factors as resilience and robustness etc. However, this work clearly indicates the potential advantages of using a relatively complex agent organization for this type of applications.

There are many possible directions to pursue with this work:(1) to explore mechanisms to encourage cooperation in the situations that agents are self-interested. (2) to explore the possibility of more sophiscated topology variations, (3) to explore self-adaptive algorithms for the agents so that they can adjust the agent-view dynamically over repeated query processing sessions. This self-adaptation will in large part be related to learning key parameter settings that in this paper were set by hand. We believe that this evolutionary approach would allow the system to adapt in an open environment, and thereby providing more stable system performance.

8. RELATED WORK

Peer-to-peer (P2P) systems have emerged as a popular way to share huge volumes of data. However, early work in this area focused on file-sharing system with exact-match based searching approach. Most recently, P2P based information retrieval has attracted tremendous attentions [4, 8, 2]. Among this line, pSearch [8] distributes document indices through the P2P network based on document semantics generated by Latent Semantic Indexing (LSI); Crespo [2] proposed a semantic overlay network (SON) to split the network into several soft clusters. However, the authors made no effort to improve the search algorithm and mainly focused on the file-sharing system without considering search aspect. Lu presented several langauge-model based search strategies in hybrid P2P network[4]. However, in this work, the authors assume the topology is already in place. Our work differs from these works in that we proposed a multi-level hiearchical group formation algorithm and a corresponding searching algorithm.

9. CONCLUSION

In the multi-agent based content sharing system, the flat, peer-to-peer(P2P) organization hinders agents from efficiently locating relevant agents for queries. This paper develops and analyzes a hierarchical agent group formation protocol to build a hybrid organization for large-scale content sharing system in an incremental and distributed fashion. During the group formation process, agents are classified into various semantic groups by organizing their agent view structure. These agents in a same group cooperate together to provide information service on certain topics. A collection model based coordinated search algorithm is also proposed

to take advantage of the organization. We evaluated the system performance based on TREC-VLC-921 datasets. The results of the experiments demonstrate a significant increase in the cumulative recall ratio(CRR) measure compared to the flat agent organization and structure.

10. REFERENCES

[1] J. Callan. *Distributed information retrieval.* Kluwer Academic Publishers, Reading, Massachusetts, 2000.

[2] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. In *Technical report, Computer Science Department, Stanford University, October 2002.*, 2003.

[3] D. Hawking, N. Craswell, and P. Thistlewaite. Overview of trec-6 very large collection track. In *In Proceedings of the Tenth Text Retrieval Conference TREC*, pages 93–105, 1997.

[4] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 199–206. ACM Press, 2003.

[5] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *In the Tenth Text Retrieval Conference, TREC 2001. NIST Special Publication*, pages 103–108, 2001.

[6] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM '2000*, November 2000.

[7] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM Press, 1998.

[8] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 175–186. ACM Press, 2003.

[9] H. Zhang, W. Croft, B. Levine, and V. Lesser. A multi-agent approach for peer-to-peer information retrieval. In *Proceedings of AAMAS 2004*, July 2004.