Balancing Ontological and Operational Factors in Refining Multiagent Neighborhoods

Leen-Kiat Soh and Chao Chen
Computer Science and Engineering
University of Nebraska-Lincoln
256 Avery Hall, Lincoln, NE 66588-0115
{lksoh, cchen}@cse.unl.edu

Abstract

In this paper, we present our work balancing ontological and operational factors in building collaborations within multiagent neighborhoods. This innovation takes into account the desired level of performance, service priorities, and relaying of tasks to determine whether an agent should entertain ontological learning, which are more expensive but more rewarding in the long run, or carry out operational learning, which are less expensive and more rewarding in the short term. The domain of application is multiagent, distributed information retrieval, where agents, safeguarding information or data resources, improve their local services by collaborating with others. Each agent is capable of providing query services to its users, and is equipped with an ontology defining the concepts that it knows and the associated documents. When collaborating, an agent needs to determine which agents to approach and how to approach them. Experiments show that with balanced profile-based reinforcement learning (operational) and inference-based ontological learning, agents reach desired level of performance while improving the neighborhood health and communication cost.

1. Introduction

In information retrieval, large ontologies are usually so diverse that they are best designed and maintained in a distributed manner by multiple experts (McGuinness 2002). Ideally, if all parties have a common vocabulary to express their ontology, then knowledge and ontology can be shared seamlessly. However, such a vocabulary is difficult to establish as different users or entities have their own ontological interpretations (Williams 2004). As a result, agents need to learn to understand each other when collaborating. Thus far, most distributed information retrieval (DIR) research have focused on improving ontology understanding among agents based on ontologies alone (e.g., Takaai et al. 1997, Bayardo et al. 1998, Williams 2004, Mine *et al*. 2004, Zhang *et al*. 2004), without taking into account the operational factors such as the number of threads available for collaboration, the helpfulness of the agents in addition to their expertise, the desired performance level of the system, and so on.

Our research takes into account both ontological and operational factors. The reasons for considering operational factors are two-fold. First, the quality of retrieval hinges upon multiple variables such as the relevance of the retrieved documents and the speed of the retrieval. The tradeoffs among between variables depend on the needs of the user. If a user prefers speed over accuracy, then the user may want a system that could return good enough results quickly. Second, for scalability, it is costly for an agent to discover the best agents in its community that could provide documents or coverage for each concept or term in the agent's ontology, especially when the system is large. Third, learning about ontologies and finding translations are expensive. It is thus wise for an agent *A* to realize first whether it needs to learn about another agent *B*'s ontology, and then determine whether *B* is able to help operationally. If *B* is always busy and not available to provide help, then learning about *B*'s ontology is not rewarding to *A* operationally, even though such learning would enhance *A*'s ontological knowledge.

Our work's underlying framework is described in (Soh 2003), in which each agent safeguards its own information or data resources and manages its local services by collaborating with others. Each agent has both operational and ontological components. When an agent intends to ask for a particular service from another agent, it may approach (1) an agent that is very capable of performing the service, or (2) an agent that is very helpful though not very capable, or (3) a help-

AAMAS'05, July 2529, 2005, Utrecht, Netherlands.

ful agent with good capability. Thus the agent may decide to learn ontologically—exchange ontological concepts and knowledge with another agent—*only if* such learning leads to operational efficiency. In this paper, we focus on how an agent refines its collaboration with its neighboring agents to satisfy queries, taking into account the desired level of performance of the system, the resources available, the frequency of queries, and the helpfulness of the neighbors.

2. Related Work

Our research work is particularly related to three research projects. Here we briefly describe them and distinguish the differences between these projects and our approach. The key difference is that these systems do not incorporate operational factors in ontological understanding and query distribution among agents.

In DOGGIE (Williams 2004), the distributed ontology understanding among agents is carried out in three steps: locating similar semantic concepts, translating semantic concept and learning key missing attributes. To locate similar semantic concepts, an agent sends other agents the name of the concept and a sample of semantic objects of that concept. The receiving agent interprets the semantics by comparing the concept and objects and then sends back the result. In essence, DOGGIE agents are able to teach each other what their concepts mean using their own conceptualization. Our work uses the same principle that allows agents to exchange ontology understanding by multiple 1-to-1 collaborations. However, our approach considers operational factors that prevent unnecessary ontological learning from taking place.

Mine et al. (2004) propose an agent community architecture that performs peer-to-peer information retrieval with three types of agents: User Interface (UI) Agents, Information Retrieval (IR) Agents and History Management (HM) Agents. A UI-agent is responsible for collecting user's query. An IR-agent is responsible for query retrieval and the communication with other agents in the community. An HM-agent is responsible for updating a pair of history: (a) a query-retrieved document history, and (b) a query-sender agent history. In our approach, each agent has the ability to interface with the user, retrieve information, and manage history of other neighbors. Moreover, Mine et al. (2004) do not consider diverse ontologies.

Zhang et al. (2004) propose another peer-to-peer information retrieval system. The agents apply an agent-view reorganization algorithm to form a local view of what other agents know and information clusters. The agents select a coalition to collaboratively share queries based on the local view. After receiving a query, an agent uses a gradient search scheme to identify the best coalition and distribute the queries to the identified coalition. When an agent is not able to locate a useful local view, it automatically forwards the query to high-degree connective agents, allowing the query to jump out of a "bad zone" to a likely "good zone." Similarly, our agents use reinforcement learning for an agent to find a "good zone" by filtering out incapable and non-helpful neighbors. However, faced with an unknown query, our agent is able to *relay* the query to a "good zone" using the recipient agent's profile of its neighborhood, to the neighbor that has been known to be useful (ontologically) and helpful (operationally).

Our work extends the results and design of (Chen and Soh 2004). Here we briefly summarize the results in (Chen and Soh 2004). First, collaboration among agents greatly improves their query services for the users. Second, learning allows agents to improve their performance over time from observing their operational neighborhoods. Third, the operational infrastructure facilitates some rudimentary ontology inference. Our current focus investigates how agents decide among two types of learning to balance its activities such that the performance of the system is maintained at a desired level.

3. Framework and Methodology

Our framework is based on (Soh 2003) where agents interact and collaborate to process and satisfy queries. Each agent has operational and ontological components designed for: operational and ontological.

The operational component is domain independent, addressing issues such as the appropriate number of threads, the response behavior of neighbors, and so on, for an agent to maintain a certain level of performance. The operational knowledge for collaboration is stored in a *neighborhood profile*. This is based on the relationship between the agent and its neighboring agents and the agent's current view of its resources. We will discuss this further in Section 3.1.

The ontological component is domain-specific, allowing the agent to satisfy its user's demands (i.e., queries) and exchange the content (mapping of ontologies) between the agent and its neighboring agents. An agent's ontology repository and *translation table* constitute its ontological knowledge. An agent can also initiate a collaboration process to learn about other ontologies. Given information from the collaboration, it infers the mapping between two concepts of different ontologies, and saves the mapping in the translation table dynamically. An agent's ontology repository consists of a set of concepts, with each concept supported by a set of documents. Two agents learn about the mapping between their concepts by comparing the associated documents. We will discuss this further in Section 3.2.

When an agent receives a query from its user, it checks the concept against its own ontology repository. If the agent finds a match *and* there are enough documents to satisfy the user, then it needs no collaboration from other agents and returns the result to the user directly. If the agent cannot satisfy the query on

its own, it will contact its neighbors for collaboration. There are three possible scenarios. First, if the agent recognizes the concept in its ontology base but does not have enough documents to fulfill the query, then it will approach its neighbors to ask for more documents. How many documents to ask from each neighbor is based on the agent's perception of the neighbor: a *collaboration utility* and a *credibility score*. Second, if the agent does not recognize the concept, then it will check its translation table and see whether the concept in the query matches (in terms of its name) any entries in the table. If found, the agent forms and relays a new query to the corresponding neighbor. Third, if the agent does not recognize the queried concept and does not believe that other neighbors know the concept, it will simply distribute the query to all its neighbors in an order derived from the collaboration utility values and past *relay* scores. An agent's relay score of one of its neighbors is the average search ratio of all the queries relayed to that neighbor.

When an agent receives a request, it checks for the request type. First, there are *collaboration* requests to supply relevant documents to a particular query. The agent will check its translation table and, depending on a successful match, retrieves the required number of documents to return to the initiator. Second, there are *inference* requests to provide ontological similarity mapping. Together with an inference request is a list of supplemental documents. The agent checks its own ontology repository to compute the similarity between its own documents and the supplemental ones, and returns the best-matched concept.

A query collaboration service is less expensive computationally and rewarding in the short term. However, an inference service is expensive computationally and only rewarding in the long run. Further, with better ontological understanding among the agents, the system as a whole will retrieve documents that are more relevant. However, when the system is resource-constrained and time-constrained, a trade-off exists such that agents may decide to learn *only sufficiently* about other agents' ontologies as long as each believes that it is achieving the desired performance level of the system.

In our design, each agent has both inference-based ontological learning and profile-based reinforcement learning. Via reinforcement learning, an agent is more likely to contact neighbors that have been helpful. Via ontological learning, an agent is more likely to approach neighbors that are considered knowledgeable.

3.1. Collaboration Utility

Our collaboration utility is based on negotiation-based parameters introduced in (Soh and Tsatsoulis 2001). We define the collaboration utility of a neighbor as perceived by an agent as the average of (a) *_helpRate*, the ratio of successful collaborations when the agent receives a request from the neighbor over the total number of requests from the neighbor to the agent, (b) *_successRate*, the ratio of successful collaborations when the agent initiates a request to the neighbor over the number of total requests from the agent to the neighbor, (c) *_nowCollaborating*, a Boolean indicator as to whether the agent and the neighbor are currently collaborating on another task, (d) *_requestToRate*, the ratio of the total number of requests from the agent to the neighbor over the total number of all requests from the agent, indicating the reliance of the agent on the neighbor, and (e) *_requestFromRate*, the ratio of the total number of requests from the neighbor to the agent over the total number of all requests from the neighbor to the agent, indicating the reliance of the neighbor on the agent. The collaboration utility is:

$$\textit{Collaboration Utility} = (\textit{\_successRate} + \textit{\_helpRate} + \textit{\_requestToRate} + \textit{\_requestFromRate} + (1 - \textit{\_nowCollaborating}))/5.$$

With the above score, we see that if an agent has been in close relationship with a neighbor, then the neighbor's collaboration utility is high. That the agent is not currently collaborating with the neighbor adds to the utility as well.

3.2. Ontology Repository and Credibility Score

As previously discussed, we describe each concept with a set of descriptors. In our framework, we use a single phrase to represent a concept and use WWW links as the descriptors. These concepts together with their links form an agent's ontology. The initial repository was built based on the WWW bookmarks of several students, where each bookmark title was used as the concept name, and the links filed under a bookmark were retrieved as the associated documents.

To computer the relevance between two documents, we use the vector-based cross product common in information retrieval (Baeza and Ribeiro 1999). The credibility of a translation between two concept names is thus the average relevance between the two sets of associated documents.

A translation table is agent-specific. It has C rows where C is the number of concepts in the agent's ontology repository. It has N columns for the N neighbors. Each entry is the corresponding concept name in a neighbor for a particular concept name that the agent knows and the translation credibility. There are entries that are NIL indicating an empty translation.

3.3. Balancing Ontological and Operational Factors

When an agent realizes that its translation table is poor, then it has the motivation to perform ontological inferences to learn more about its neighbors' ontologies. For each concept, it will attempt to resolve the least credible translation first. It does this by initiating an inference service request hoping that the responding neighbor will provide a mapping.

For the responding agent, the inference process is time consuming since it involves retrieving documents, extracting keywords, and comparing among numerous documents of different concepts. Such a process costs thread resources and computation, especially when the ontology repository of the responding agent is large.

Because of the limited resources, an agent has to regulate its inference processes. It must balance between *knowing more ontologically* and *providing good enough services* to its users. Our design discourages an agent from initiating too many inference processes in the following manner. First, when the responding agent, after the inference process, finds the credibility value to be very low (lower than a pre-defined threshold), then both the agents will remember it. It is more likely for the responding agent to entertain another inference request from the initiating agent in the future. Thus, the initiating agent has to choose carefully which neighbors to approach. Second, if the past relationship between two agents has not been good, then (1) it is more likely for the agents to have nothing in common in terms of concepts, and (2) even if they do, it is unlikely for them to help each other via the query collaboration requests due to the operational issues.

Therefore, if an agent focuses too much of its effort on ontological inferences, it might not have enough resources to handle the actual queries from its users. Thus, the ontological inferences and the query satisfaction tasks could benefit each other as well as detract from each other. Both improve the system and agent's performance but both also compete for resources and neither can be dominating at the same time. The challenge is to find a *balanced* level so that both work relative well together to achieve the desired level of performance.

Desired Level of Performance. In our design, we use a desired level of performance to help guide the agents in their balancing act. For example, if the system is expected to perform at a 60% success rate, then each agent will try to reach that level by learning about its neighbors if it does not have enough knowledge to achieve that success rate; if the system performs above that success rate, each agent will reduce its workload (i.e., the number of neighbors approached for help), thus reducing the message traffic and computations.

Priority. Since inference is costly, the translation table should only be improved gradually and selectively (Chen and Soh 2004). For example, when an agent tries to decide when to ask for the translation of a certain concept, it should decide based on how well the queries for that concept have been satisfied. If it has been successful, the motivation to ask for a translation is low. An agent's ability to evaluate the incoming requests and to select the most important tasks to perform becomes crucial. It should refuse some of the query collaboration requests if they do not add to the goal of the system. Towards this end, each agent keeps track of *priority* values of ontological learning and query collaboration. When an agent is performing poorly in satisfying its own queries and its translation table is not credible, it increases its priority for ontological learning. On the other hand, when the agent is performing well in satisfying its own queries, it becomes more altruistic and increases its priority for query collaboration. Each agent sets its priority based on its observation of its performance in the past W = 10 cycles.

Relays. Finally, relays occur when an agent does not recognize a query. When the agent does not recognize the queried concept, it checks its translation table to find any matched entries. If a match is found, the agent knows that one of its neighbors is likely to be able to answer the query. Thus, it forms and relays a new query to the corresponding neighbor. We call this type of collaboration a "*targeted relay*". However, if the agent does not recognize the queried concept *and* does not believe that other neighbors know the concept, it will simply distribute the query to all its neighbors in an order according to the collaboration utility scores of its neighbors. We call this type of collaboration a "*generic relay*".

4. Experiments

We have conducted a comprehensive set of experiments to investigate the ontology inference and operational efficiency. Due to the page limit, we will report on the changes in performance as agents learn to collaborate while adjusting to the desired level of system performance, and the impact of learning on the health of the neighborhood and the communication cost. For a detailed treatment of all the experimental results on incorporating ontological and operational factors, please refer to (Chen 2004).

4.1. Experimental Setup

In our experiments, we setup a multiagent system with 5 agents and 5 simulated software users. Each agent is paired up with a particular software user—the agent receives queries from the software user periodically. Initially, each agent's neighborhood consists of all other agents. That means, each agent is able to communicate with all other agents directly. For different experiments, an agent may have 5, 10, 15, and 25 collaboration threads. The results reported here are based on the configuration where an agent had only 10 such threads.

The agents' ontology repositories are heterogeneous with different concepts. Some agents also have larger repositories than the others. This difference implies that an agent with a larger repository will expend more effort when it performs ontology inferences but can be more resourceful in terms of satisfying queries and helping other agents. We decided to use such

a setup to more closely simulate a real world environment.

Each software user has a query configuration file that submits pre-defined queries to the corresponding agent. Each query in a configuration file consists of (a) a cycle number, (b) the queried concept, (c) the number of link desired and (d) the time constraint given by the software user indicating how long the user will wait for the query result.

The queries that each agent received several types of queries: (a) queries that are known to the agent, (b) queries that are known to one of its neighbors but unknown to the agent, (c) queries that are known to one or more of its neighbors but not known to the agent, and, and (d) queries that are unknown to the entire neighborhood. The percentage of queries received that is known to an agent is a random number uniformly generated between 20.0% and 33.3%. The percentage of queries received that triggered a relay (either targeted or generic) is set roughly at 25%. These queries were arranged into a batch and the same batch was fed into the system 7 times. Each segment had 150 different total queries.

Query-Triggered Collaborations. Here we list the six collaboration types that an agent might encounter during its query satisfaction process.

- *Collaboration Type 1*: The agent knows the queried concept and has enough documents to satisfy the query alone. In this case, no collaboration is needed. The agent will answer the query alone.
- *Collaboration Type 2*: The agent knows the queried concept but does not have enough documents to satisfy this query. It has some idle threads. It will use the translation table and neighborhood profile to rank the neighbors and distribute the remaining number of requested documents among its neighbors based on the weighted sum of both the collaboration utility and the translation credibility of the individual neighbors.
- *Collaboration Type 3*: Similar to Type 2, but the agent has *no* idle threads and returns whatever documents that it has immediately.
- *Collaboration Type 4*: The agent does not know the queried concept. The agent has no idle threads and directly terminates the query process.
- *Collaboration Type 5*: The agent does not know the concept. It has some idle threads. It discovers that one of its neighbors knows this query by checking its translation table. The agent will relay the query to that specific neighbor and record the acceptance and satisfaction quality of the neighbor. This is a targeted relay.
- *Collaboration Type 6*: The agent does not know the queried concept. It has some idle threads. However, it cannot find any neighbor that might know this concept. The agent will distribute the request among all neighbors based on their collaboration utility and relay scores. This is a generic relay.

Types 3 and 4 collaborations are situations in which the agent cannot approach potentially helpful neighbors for help because it does not have available collaboration threads. Further, Types 2, 5, and 6 collaborations are situations where the agent has the resources to carry out query collaborations, indicating that it is capable operationally. A good multiagent system should reduce such the occurrences of Types 3 and 4 collaborations and increase Types 2, 5, and 6 collaborations. Reducing the numbers of Types 3 and 4 collaborations indicates that the agents are able to better utilize their resources and avoid fruitless requests for collaboration. Increasing the numbers of Types 2, 5, and 6 collaborations, on the other hand, indicate that the agents are able to identify helpful and useful neighbors.

4.2. Experimental Results

Query Frequencies and Learning Rate. We were interested in how the profile-based reinforcement learning behaved under different query frequencies. We investigated two query configurations. In the first configuration (30/30), the software user submitted thirty queries to the agent in thirty cycles of an agent, constituting one batch, and iterated this process seven times, resulting in a total of seven batches of queries. In the second configuration (30/60), the software user performed the same thing in sixty cycles. Thus, the first configuration has a higher query frequency than the second one. A higher query frequency means a more demanding load on the system. We would like to observe how learning is affected by the load of the system. We set the desired performance level, *DP*, at 0.6 (or 60% query satisfaction) for these two configurations. Note that we have carried out experiments on different performance levels (0.2, 0.4, 0.6, 0.8, and 1.0). As will be discussed later in our summary, the results with $DP = 0.6$ is representative of the results with $DP = 0.2$ and $DP = 0.4$.

Figures 1-3 show, respectively, the average response time per batch, the average search ratio per batch, and the average number of neighbors contacted per batch in each configuration.

From Figure 1, we see that the average response time in 30/30 decreased significantly over time. It indicates that the agents learned how to satisfy queries more quickly over time under demanding load. However, in the case of 30/60, because the agents were not under the same stress, the impact of learning was not as significant. That is, all batches of queries were answered in a rather uniform, timely manner.

Figure 2 depicts the average search ratio, a metric measuring the ratio of number of documents retrieved over the number of requested documents for a query, per batch for the two frequency configuration. This measures how well an agent satisfied the queries in terms of the retrieved results. We saw that due to learning, the agents in the 30/30 configuration strived

to reach the targeted *DP* = 0.6. After three batches, they reached the target. But to clamp the improvement at 0.6, the agents over-adjusted and decreased their search ratio values. A low load with 30/60 did not see any trends towards reaching *DP*. Note that when an agent realizes that it is performing around the desired level of performance of the system, it changes its priority of services, de-emphasizing query collaborations and favoring ontological learning, as it tried to squeeze in costly ontological inference tasks when its query tasks are going well. However, these costly inferences, though gradual and selective, could still hold up critical resources such as the collaboration threads. Looking more closely at Figure 2, we see that every agent had an upward tendency towards the later portion of their runs, hinting that each might be able to reach *DP* again if the runs were extended further.

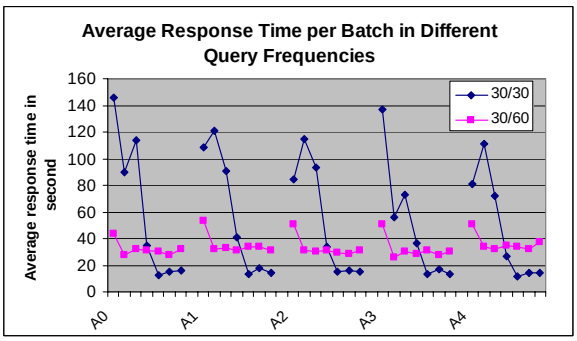

Figure 1. Average response time per batch for query frequency 30/30 and 30/60 (*DP* = 0.6).

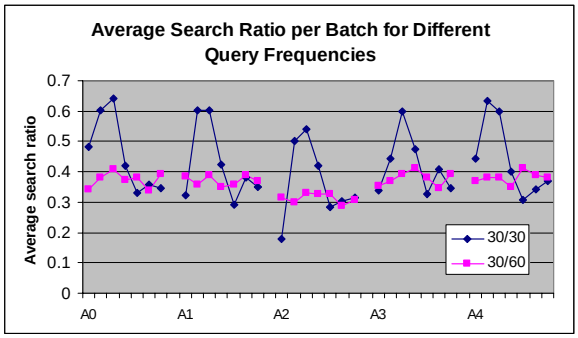

Figure 2. Average search ratio per batch for query frequency 30/30 and 30/60 (*DP* = 0.6).

Figure 3 shows the average number of neighbors contacted per batch for the two query configurations. We see that the agents were able to reduce the number of neighbors contacted. Compared to Figures 1 and 2, we see that the agents were able to improve their response time, the search ratio, and the numbers of neighbors contacted at the same time, around the third and fourth batches of queries.

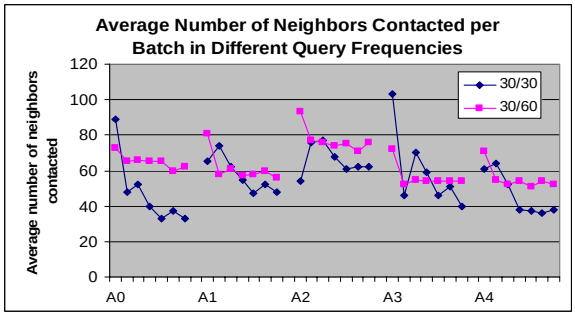

Figure 3. Average number of neighbors contacted per batch for query frequency 30/30 and 30/60 (*DP* = 0.6).

We conclude that the learning rate is much more significant when the load is more demanding (30/30 vs. 30/60). We also observe that profile-based reinforcement learning is adaptive to the circumstance that the agents are in. If the resources are abundant and the user's demand is not high, then the need for learning is low and will not be carried out as often as the agents will enjoy a rather good level of performance. If the resources are highly constrained and the user's demand is high, the system will not do as well and thus agents are motivated to learn. Not shown in this paper are the neighborhood response rate and on time rate, which measures the percentage of neighbors responding to a query help request and the percentage of times a neighbor returns the requested documents on time, respectively. Both these rates improve. That is, the agents are able to contact fewer neighbors but with neighbors that are more helpful and useful. As a result, the queries can be answered on time more often and with better recalls (higher search ratios).

Neighborhood Health. We take a closer look at the improvements incurred in the neighborhood of each agent. The neighborhood health is a composite index that indicates the quality of the neighborhood of an agent. The health of an agent's neighborhood is a weighted sum of three parameters: (1) the average credibility score stored in the translation table of the agent counting only the neighbors approached, (2) the average collaboration utility of the neighbors approached, and (3) the average relay score of the neighbors approached for relaying. Each neighbor approached will contribute at most 3 points to the quality of the neighborhood health. So the maximum possible quality of health in this case is 12 for each agent since each has 4 neighbors.

Figure 4 shows the neighborhood health as the agents gain knowledge in translation and expertise in collaboration through inference-based ontological learning and profile-based reinforcement learning. It shows that each agent is able to refine its neighborhood gradually and converge. Indeed, though each agent does not improve its own neighborhood significantly beyond a certain point, each was able to form different collaborations for different queries, contacting only a subset of neighbors for each query. As a result, each contacted about 2 neighbors after convergence, with a "good enough" translation table to achieve *DP* = 0.6.

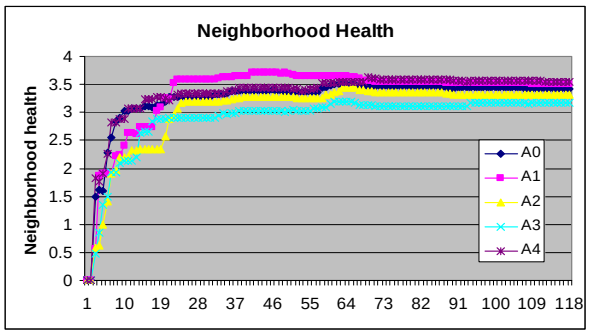

Figure 4. The health of an agent's neighborhood agents over the number of queries seen (*DP* = 0.6).

Communication Cost. Our goal is to reduce the number of messages sent and received by every agent in the system while maintaining the quality of services. We observe that, in general, the numbers of messages sent and received by every agent decreased by around 25-100 messages per agent between the peak and the last measure, as shown in Figure 5. In the beginning, though the agents were contacting fewer neighbors for query collaborations, some of them also requested ontological inference services. That caused the number of messages to stay high for a while. The impact of learning on the communication cost was actually felt during the third or fourth batch of the experiments.

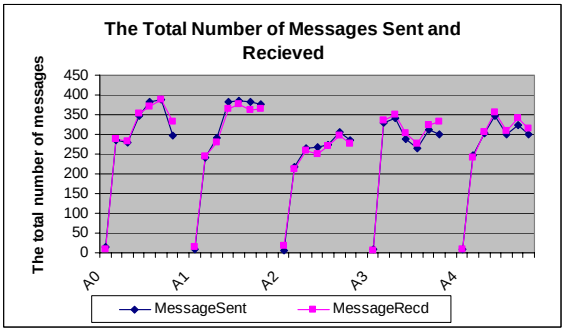

Figure 5. The total number of message sent and received for all agents per batch over time.

Collaboration Types. Figure 6 shows the numbers of different types of collaborations in each batch for the five agents. As learning progresses over time, the number of Type 5 collaborations (targeted relays) increased because the agents gradually learned what the other agents knew and what itself did not know through ontological inferencing. Further, the number of Type 6 collaborations (generic relays) decreased because the agents became more knowledgeable about the other agents' ontologies. Thus, the agents became more responsible in asking for help—less "spamming". The number of Type 2 collaborations remained the same as the local ontology repository of each agent did not change. Best of all, the numbers of Types 3 and 4 collaborations (situations where no idle threads were available for collaborations) significantly decreased. This indicates that the agents were able to learn to use their resources effectively. Combining Figure 6 and other figures, we see that the agents, while reducing their use of the resources (bandwidth and collaboration threads), were still able to achieve good neighborhood health and better performance parameters while adapting to the desired level of performance.

Not shown in the figures in this paper are the performance measures of Types 2, 5, and 6 collaborations. Here we briefly report on them. The performance of Types 2 and 5 collaborations were significantly improved by profile-based reinforcement learning. In Type 5 collaborations (targeted relays), we observe that the agents were able to identify unknown queries and relay the queries to appropriate neighbors such that the search ratio improved. However, in Type 6 collaborations (generic relays), the agents needed the relay score in addition to the collaboration utility to obtain improved performance. This indicates that even when an agent had absolutely no idea about which neighbor knew about a particular queried concept, it was still able to improve its performance by looking at two operational factors: the collaboration profile and the relay score, with the latter keeping track of the response of a neighbor to a relay request.

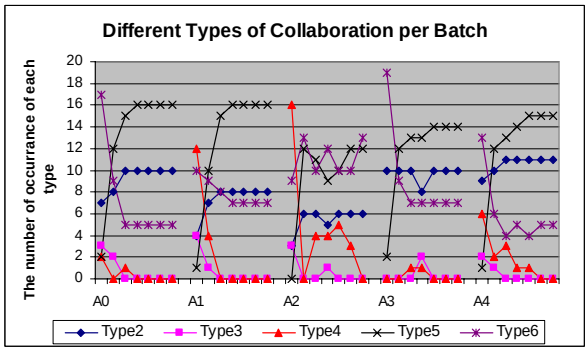

Figure 6. The number of occurrences of different types of collaborations over time.

Summary and Discussions. Profile-based reinforcement learning is important for the agents to make good decisions based on what they observe in the past. The operational knowledge (collaboration utility and relay scores) stored in the neighborhood profile provides good information of the probability of a neighbor accepting a query collaboration request. Combining both the operational and ontological knowledge, the agents learn to select the most helpful and capable neighbors for collaboration. As a result, the quality of collaboration improves. The learning is effective in reducing the average response time, improving the quality of query satisfaction, and reducing the number of neighbors contacted and the communication cost.

There are also indications that our learning and adaptive mechanisms are able to adapt to a desired

level of performance. When the desired level of performance is too high (> 0.6) such that the system does not have the ontological resources (documents) to sustain that type of performance, the agents will go into an *overdrive* to try to match it. In that case, the agents were observed to constantly conduct ontological inferences; the requests among the agents were mostly for ontological inference services as those became the priority of the agents. The system's performance thus suffered. On the other hand, when the desired level of performance is low (≤0.6), the agents were observed to be able to adapt rather well, reducing their response time, the number of neighbors used, etc., while nearing the desired level of performance.

Our ontology repository for the experiments is still simplistic. There is no hierarchical relationship within our ontologies. In real-world applications, ontologies are usually organized into hierarchies and there are many relations among the concepts such as super class and sub-class relation, equivalent relation, *is-a* and *has-a* relations, etc. We are also designing a recommender module in each agent based on the relay scores of its neighbors. With this module, we aim to allow a user to be directly routed to the responsive neighbors for certain queries that the agent does not recognize.

5. Conclusions

We have presented a multiagent, distributed information retrieval system in which collaborating agents improves their performance by learning ontologically and operationally. This paper investigates the tradeoffs between ontological and operational factors in refining multiagent neighborhoods. We have described the use of collaboration utility and translation credibility, the adoption of a desired performance level to tradeoff between the operational and ontological activities, the dynamic determination of service priority based on agent observation of past performance, and the use of generic and targeted relays. We have reported on profile-based reinforcement learning, inference-based ontological learning, and observation-based priority determination. Our experiments have shown that our "balanced" approach was able to improve the quality of the collaborations in terms of the response time, quality of the retrieved results, the number of neighbors contacted, the number of messages sent, and the neighborhood health.

6. References

Baeza, R. and B. Ribeiro (1999). *Modern Information Retrieval,* The ACM Press, Addison Wesley.

Bayardo, R., W. Bohrer, R. Brice, et al. (1998). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments, In *Readings in Agents*, M. Huhns and M. Singh (Eds.), San Francisco: Morgan Kaufmann, 205-216.

Chen, C. (2004). A Multiagent Approach Using Ontology and Operational Learning to Improve Distributed Information Retrieval, M.S. Thesis, University of Nebraska, Lincoln, NE.

Chen, C. and L.-K., Soh (2004). Adaptive Learning to Optimize Resource Management in a Multiagent Framework, *Proc. ICAI'2004* Las Vegas, NV, pp. 386-389.

McGuinness, D. L. (2002). Conceptual Modeling for Distributed Ontology Environments, *Proc. 8th Int. Conf. Conceptual Structures Logical, Linguistic, and Computational Issues,* Darmstadt, Germany, August.

Mine T., Matsuno D., Takaki K., and M. Amamiya (2004). Agent Community based Peer-to-Peer Information Retrieval, *Proc. AAMAS'2004,* July 19-23, NY, pp. 1484-1485.

Soh, L.-K. (2003). Collaborative Understanding of Distribute Ontologies in a Multiagent Framework: Design and Experiments, *Proc. AAMAS 2003 Workshop OAS,* Melbourne, Australia, pp. 47-54.

Soh, L.-K. and C. Tsatsoulis (2001). Reflective Negotiating Agents for Real-Time Multisensor Target Tracking, in *Proc. IJCAI'01,* August 6-11, Seattle, WA, pp. 1121-1127.

Takaai, M., H. Takeda, and T. Nishida (1997). Distributed Ontology Development Environment for Multi-Agent Systems, *Working Notes AAAI-97 Spring Symp. Series on Ontological Engr.*, pp. 149-153.

Williams, B. A. (2004). Learning to Share Meaning in a Multi-Agent System, *J. Autonomous Agents & Multiagent Systems*, 8(1):165-193.

Zhang, H., Croft, W. B., Levine, B., and V. Lesser (2004). A Multiagent Approach for Peer-to-Peer Information Retrieval, *Proc. AAMAS'2004*, July 19-23, NY, pp. 456-463.