Task-based Multi-agent Coordination for Information Gathering

Katia Sycara and **Dajun Zeng**
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
(412) 268-8825 (412) 268-8815
katia@cs.cmu.edu zeng+@cs.cmu.edu

Introduction

The ubiquity of network-based information resources has given impetus for the development of intelligent software agents that will be able to (1) gather task-related information automatically or with little help from human users from various on-line information resources, (2) resolve potential conflicts among acquired knowledge from different information resources, and (3) more importantly, collectively and effectively solve tasks requested by human users.

We report on our work on developing distributed collections of intelligent information agents that cooperate asynchronously to perform goal-directed information retrieval and information integration in support of various tasks, such as finding information about people on the Internet, managing calendars and making arrangements to host visitors in an academic environment. The task of hosting a visitor involves arranging the visitor's schedule with faculty that match the interests that the visitor has expressed in his/her visit request. The visitor hosting task is one of the tasks that are investigated in the context of the PLEIADES project at Carnegie Mellon University. The broader goal of PLEIADES is to characterize and develop distributed agent-based architectures that are composed of *negotiating and learning agents* and apply them to tackle information and activity management problems for everyday use. Agents coordinate and negotiate with each other to resolve disparities in the retrieved information. In addition, they learn from their users, the information sources, and each other.

In this paper, we will focus on the architecture of our distributed system, and the interactions among its agents for task-based accessing of heterogeneous, distributed information sources. We will use the visitor hosting task as an example domain for illustration.

Architecture for Cooperative Intelligent Information Retrieval

In the collection of agents we have developed, we distinguish two types of agents: *task-specific* software agents help users perform tasks by communicating with each other and/or querying and exchanging information with *information-specific* software agents, which provide intelligent access to a heterogeneous collection of information resources. For example, the meeting scheduling module could be a task-specific agent, which will manage and update a particular user's appointment and meeting agenda. The general-purposed `finger` service module, which can extract useful information from the network `finger` utility given user's login name and network address, can be viewed as an information-specific software agent. Although the boundary between these two types of software agents is quite arbitrary and vague, we make the distinction that typically task-specific agents access other agents (either task-specific or information-specific ones) and communicate directly with users, whereas information agents (usually) access only information sources, other information assistants and task-specific agents. Task-specific agents have knowledge of the task domain, information assistants relevant to performing various parts of the task. In addition, task assistants have strategies for resolving conflicts and fusing information retrieved by information assistants. Information assistants have models of the associated information resources, and strategies for conflict resolution and information fusion.

This architecture is mainly motivated by the following considerations:

- *sharability:* Many users can share information-specific software agents or task-specific agents. Typically, user applications will access several agents in parallel and one software agent can serve different application programs. The behavior of a software agent, essentially, could be easily described in a *server-client* model.
- *Complexity hiding:* Often information retrieval in support of a task involves quite complex coordination of many different agents. Having the user interact only through a relevant task assistant hides the underlying complexity and frees the user from having to know of, access and interact with a plethora of information seeking agents in support of a task. For example, the hosting visitor task in-

volves four information agents and four task agents. However, the user interacts directly only with the `Visitor Hoster` agent, the main task assistant for the visitor hosting task.

- *modularity and reuseability:* Although software agents will be operating on behalf of their *patrons*—human users, pieces of code can be copied from one user to another without modifications or with little adaptation to take into consideration particular users' preferences or idiosyncrasies. One of the basic ideas behind the distributed agent-based approach is that software agents will be kept simple for ease of maintenance, initialization and customization.
- *flexibility:* software agents can interact in new configurations "on-demand", depending on the information requirements of a particular decision making task.

Scenario: Organize a Visit

We illustrate the system architecture and the interactions of the information gathering agents in the visitor hosting scenario. A different variation of the hosting visitor task has also been explored by Kautz and his colleagues at Bell Labs (Kautz, Selman, & Coen 1994).

A visitor hosting agent should have the following capabilities:

- It should automate information retrievals in terms of finding personnel information of potential meeting attendees. It should be able to access various on-line public databases and information resources at the disposal of the visit organizer. The system should also integrate the results obtained from various databases, clarify ambiguities (e.g., the synonyms for certain entities) and resolve the conflicts which might arise from inconsistency between information resources. Some possible information resources that are common to a modern university are: networking finger, on-line library, on-line phonebook, etc.
- It should create and manage schedule for visitors. It is also preferable if the meeting location and equipment can be managed in a coherent way.
- It should possess a graphical user interface which can interact with the users. The GUI facilitates getting input from the user, presenting acquired information, asking for user confirmation as well as advising the user of the state of the system and its progress.

Our prototype system retrieves information from various heterogeneous information resources at $\mathcal{CMU}$ and also internet-based resources, such as remotely accessing plan files at sites external to $\mathcal{CMU}$ to extract information about people. The currently implemented agents that are utilized in support of hosting visitors are:

- Information-Specific Agents
  1. `Finger` agent, which heuristically parses the retrieved information from remotely residing `finger` data bases. The possible types of information that can be acquired in this way include: work title, research interests, work and home phone numbers, vacation plan, etc.
  2. `Who's-Who` agent, which accesses on-line $\mathcal{CMU}$ who's who database through http-based queries. The fields in the database include: name, title, affiliation, campus office, campus phone number, home address and E-mail address.
  3. `Faculty Interests` agent, which can be used to retrieve information about the faculty members in the School of Computer Science at $\mathcal{CMU}$ with respect to their research interests.
  4. `Computer-Science-Directory` agent, which can get the information about phone number, office number, home address, etc. for all the members of the School of Computer Science at $\mathcal{CMU}$, including faculty members, staffs and students.
- Task-Specific Agents
  1. `Visitor Hoster` agent, which accepts input from the user concerning the information about the visitor and intended specification of possible meeting candidates, and initiates other related personnel information agents and scheduling agents.
  2. `Scheduling` agent, which takes the responsibility of maintaining a visitor's meeting schedule, coordinating among different meeting requests meanwhile taking into consideration possible meeting preferences of meeting attendees. For example, user A might prefer to meet with the visitor in the afternoon although meeting in the morning is also admissible.
  3. `Personnel Finder` agent, which coordinates all personnel information agents through a `DataBase-Mapper`, in which mapping functionality from available knowledge to information assistants containing desired information is provided. After answers from information agents get collected, `Personnel Finder` will try to resolve conflict heuristically [1] and merge them together to get a coherent picture about meeting candidates. In addition, the `Personnel Finder` agent, given a person's name and affiliated organization, has heuristics for composing the person's e-mail address and remotely accesses through internet `Finger` information pertaining to the person.
  4. `Interface` agent, which takes care of presenting acquired information from task or information specific agents to human users. It also handles the

[1] One of the heuristic rules we are using is that if the information returned by `Finger` is different from what `Computer-Science-Directory` found, we assume that the information based on `Finger` is more relevant and up-to-date.

input from users. Separating interface functionalities from agent functionality helps increase the system modularity and makes it possible to enhance human-computer interface without affecting other parts of the system.

We briefly present a visitor hosting scenario to illustrate the interactions of the various agents. Suppose Gio Wiederhold wants to visit $\mathcal{CMU}$ CS department. Gio has requested that he would prefer to meet with $\mathcal{CMU}$ faculty interested in data bases. Relevant information about Gio, such as first and last name, affiliated organization, date and duration of his visit and his preference as to the interests of faculty he wants to meet with are input into the `Visitor Hoster` agent. Then the `Personnel Finder` agent gets invoked and first accesses the `Faculty Interests` information specific agent to get a list of potential meeting candidates whose research interests match Gio's preference. Based on the list of names returned in answer to this information gathering query, the `Personnel Finder` tries to collect more information for these potential meeting candidates so that they can be contacted and asked about whether they would be interested in meeting with Gio. For each potential meeting attendee, the `Personal Finder` agent spawns multiple queries to various information specific agents, i.e., the `Finger` agent, the `Who's-Who` agent and the `Computer-Science-Directory` agent simultaneously. These agents in turn format the queries in accordance with the format of the corresponding information resource. In particular, information is gathered about the faculty rank, office location, telephone and e-mail address of each of the potential meeting attendees. The `Personnel Finder` agent receives the replies for each potential attendee, merges the information and resolves any conflicts. It then sends the information to the `Visitor Hoster` agent. The `Visitor Hoster` agent selects the e-mail addresses of the most senior faculty [2] and automatically sends them e-mail asking if they would like to meet with Gio on the date of his visit. If faculty members have personalized calendar management assistants (e.g., Tom Mitchell's Calendar Apprentice CAP (Dent *et al.* 1992; Mitchell *et al.* 1994)), the `Visitor Hoster` agent communicates with those assistants. For those faculty members who do not have a software calendar manager, the e-mail is in human readable form. Upon receipt of answers as to which faculty is interested in meeting with the visitor, `Visitor Hoster` starts its scheduling agent and works out a feasible schedule with the help from involved meeting candidates' software agents, e.g., through exchanging appointment agenda and personal calendar information.

There are some interesting features in our approach that deserve being mentioned here: (1) Information specific agents have a term-translation capabilities that processes the retrieved data to disambiguate the information as well as standardize and transform the keys to facilitate information comparison and integration from different sources. For example, the phone number prefix for $\mathcal{CMU}$ 412-268 is automatically added to $\mathcal{CMU}$ telephone extensions. Another example is that the field name *campus* used by library `Who's-Who` database is transformed to *office* — the term used in other databases. (2) In our implementation of information specific agents every separate process is handled with a timeout cap, which guarantees that a hung-up database will not hang-up the whole system. A default action gets called if the timeout is exceeded. Exception handling mechanisms could implement actions such as retry access or give up and report failure.

[2] Currently, the default duration of a meeting is one hour, so for a full day's visit, 8 faculty members are initially selected.

Research Issues and Future Work

Some of the issues that we plan to focus on in the future include (1) representing and associating with the information sources meta-information, such as size, average time it takes to answer a query and monetary cost of query processing, (2) caching answers to queries that are frequently asked and determining how to manage the cached information, (3) inductively learning data base regularities and use the learned regularities during agent interactions, (4) learning information retrieval and negotiation strategies, and (5) learning information about the capabilities and reliability of different agents.

Acknowledgments

This research has been sponsored in part by ARPA Grant F33615-93-1-1330. We want to thank Tom Mitchell, Rich Caruana, Dana Freitag, Matthew Glickman, Ken Lang, Sean Slittery, David Zabowski and other members of the PLEIADES project for interesting discussions. We also want to thank Gilad Amiri for doing much of the implementation.

References

Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zabowski, D. 1992. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI.

Kautz, H. A.; Selman, B.; and Coen, M. 1994. Bottom-up design of software agents. *Communications of the ACM* 37(7).

Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; and Zabowski, D. 1994. Experience with a learning personal assistant. *Communications of the ACM* 37(7).