MACRON: An Architecture for Multi-agent Cooperative Information Gathering

Keith Decker, Victor Lesser, M. V. Nagendra Prasad and Thomas Wagner*
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA.
{decker,lesser,nagendra,wagner}@cs.umass.edu

November 16, 1995

Abstract

The complexity of the modern information carrying landscape requires a sophisticated view in which information is *acquired* rather than simply retrieved; where the process must be dynamic, incremental, and constrained by resource limitations. In this paper, we describe a multi-agent architecture called MACRON based on the Cooperative Information Gathering (CIG) paradigm designed specifically for such complex information gathering environments. Top level user queries drive the creation of partially elaborated information gathering plans, resulting in the employment of multiple cooperative agents for the purpose of achieving goals and subgoals within those plans. Agents in MACRON incorporate capabilities to exploit subproblem interdependencies, manage uncertainty inherent in multi-agent search, intelligently trade-off solution quality for resource limitations and exploit or avoid redundancy as needed.

1 Introduction

The complexity of the modern information carrying landscape requires a sophisticated view in which information is *acquired* rather than simply retrieved; where the process must be dynamic, incremental, and constrained by resource limitations. Information Gathering (IG) is an activity involving pro-active acquisition of information, from possibly heterogeneous sources, in response to a complex query that may require the system to possess capabilities such as reasoning, representation, and inferencing. The set of data that represents the best response to a user's query may be the aggregation of data acquired from distributed, heterogeneous information sources. In addition to the complexity of query specification, control

*This work is supported in part by NSF Center for Intelligent Information Retrieval (CIIR), and Office of Naval Research contract N00014-95-1-1198. The content of the information does not necessarily reflect the position or the policy of the U.S. Government and no official endorsement should be inferred.

of the acquisition process may itself be complex and dynamic in IG systems. Traditional Information Retrieval (IR) is a limited sub-case of such information gathering systems in which queries generally map onto static, pre-specified retrieval plans.

We recently presented a model of *Cooperative Information Gathering* (CIG)[13] designed specifically for such complex information gathering environments. In this paper, we describe an architecture called MACRON (**M**ulti-agent **A**rchitecture for **C**ooperative **R**etrieval **ON**line) based on the CIG paradigm for multi-agent information gathering activities. User queries drive the creation of partially elaborated information gathering plans, resulting in the employment of multiple cooperative agents for the purpose of achieving goals and subgoals within those plans. The agents use existing information resources available on the Internet such as newsgroups, archives, magazine databases, and corporate WWW sites. This process can be viewed as "distributed problem solving," where the agents are semi-autonomous, socially situated, and can react to a dynamically changing environment. For example, agents may explicitly cooperate to achieve their goals. Cooperation between agents implies management of interdependencies between their activities so as to integrate and evolve consistent clusters of high quality information from distributed heterogeneous sources. Agents may share information with one another that affect what plans they execute to achieve their sub-goals, what order they execute actions, and when they execute them. Another feature is that the agents are time-aware and free to develop different information gathering plans that depend on the amount of time they have to produce an answer. Eventually, some agents will use technologies like natural language form-filling programs and the INQUERY indexing scheme to make decisions about how to focus their search without requiring detailed user feedback.

A multi-agent approach to information gathering is promising for a variety of reasons:

- Multiple agents offer concurrency which is a big win in time constrained situations or when the search space is very large, as in the case of networked information gathering. Query plans can often be decomposed into relatively independent sub-plans with few interdependencies. An agent executing a sub-plan functions relatively autonomously but needs to coordinate with others where interdependencies exist.

- When a system is dealing with enormous quantities of data, distributed computation at the sites where the data resides may often be the more viable approach compared to migrating data to a centralized processing location. Instead of gathering data dispersed across networked information servers at a centralized site, and then evolving a coherent response to a query, agents can reside at the sources and perform distributed coordinated retrieval to prune their data space and convey a much smaller subset to the query system for further processing.

- Agent-based architectures offer modularity, robustness, separation of concerns, and other advantages of a distributed system. Information agents can be constructed and maintained separately. An agent can build upon other information agents to provide an abstraction of these information sources. In addition, passive data sources like databases can be transformed into information agents by wrapping them with intelligent interfaces[11].

The rest of the paper is organized as follows. Section 2 introduces the MACRON as a Cooperative Information Gathering organizational architecture and discusses it in detail. Section 3 discusses the status of implementation and our future plans. The final section briefly discusses related work and concludes.

2 MACRON: An Organizational Architecture for CIG

The MACRON multi-agent organization is a concrete instance of a CIG problem-solving system. MACRON incorporates capabilities to exploit subproblem interdependencies, manage the uncertainty inherent in multi-agent search, intelligently trade-off solution quality for resource limitations, and either exploit or avoid redundancy as needed. MACRON consists of an overall organizational architecture and three types of autonomous agents: DECAF reasoning agents, low level network retrieval agents, and user interface agents.

2.1 Organizational Architecture

Every multi-agent system has a (perhaps implicit) organizational architecture that sets forth the roles of the agents involved in the system. An agent's organizational role answers for the agent the questions of what classes of tasks the agent must be prepared to deal with, what resources the agent has to draw on, and what performance commitments the agent is responsible for. The MACRON organizational architecture is thus a particular set of answers to these questions, developed in response to the particular needs of CIG.

From an organizational point of view, the agents in MACRON form a matrix organization consisting of both functional and query-answering units as shown in Figure 1. In a human matrix organization, people belong long-term to a functional group (such as hardware designers, system software programmers, and applications interface programmers) and are brought together short-term into a project group (for the duration of the project). Such organizations allow a great deal of flexibility in uncertain and dynamically changing environments, while still allowing the intelligent assignment and tracking of limited resources (in MACRON, the agents).[1]

A functional unit provides access to some particular type of information resource such as current and archived news feeds, or magazine databases. Each functional unit has a functional manager (FM) agent who's role is to make task assignments to agents within the unit, given a request from a query-answering agent. Functional units may comprise undifferentiated agents that have similar knowledge and capabilities, or differentiated agents with focussed skills. A FM agent has an abstract view of the overall state of its functional unit and hence can perform task allotment intelligently, taking resource constraints and special agent capabilities into consideration[2].

[1] Human matrix organizations also have some problems, such as difficulties in achieving appropriate recognition for promotions, etc., and the unwillingness to disband project teams because of the uncertainty involved—luckily these do not (yet) seem to be problems for our autonomous computational agents.

[2] The same allocation can be achieved in a distributed manner by the agents involved through coordinated distributed scheduling.

A query-answering unit comprises a query-manager agent and a set of agents drawn from the various functional units to respond to a particular user's query. The query manager is responsible for developing an initial high-level information gathering plan, recruiting agents from the necessary functional units (via the functional unit managers), and monitoring the plan's execution (including keeping the user up-to-date and able to steer the plan as it is carried out). The user interacts with a query manager via a web browser and HTML forms.

Agents interact with one another using KQML (Knowledge Query and Manipulation Language, part of the ARPA Knowledge Sharing Effort) and a task structure specification language based on the TÆMS (Task Analysis, Environment Modeling, and Simulation) representation framework. TAEMS represents the structure of tasks (like information gathering), including the hard and soft relationships between tasks, for the purpose of coordinating and scheduling agent actions.

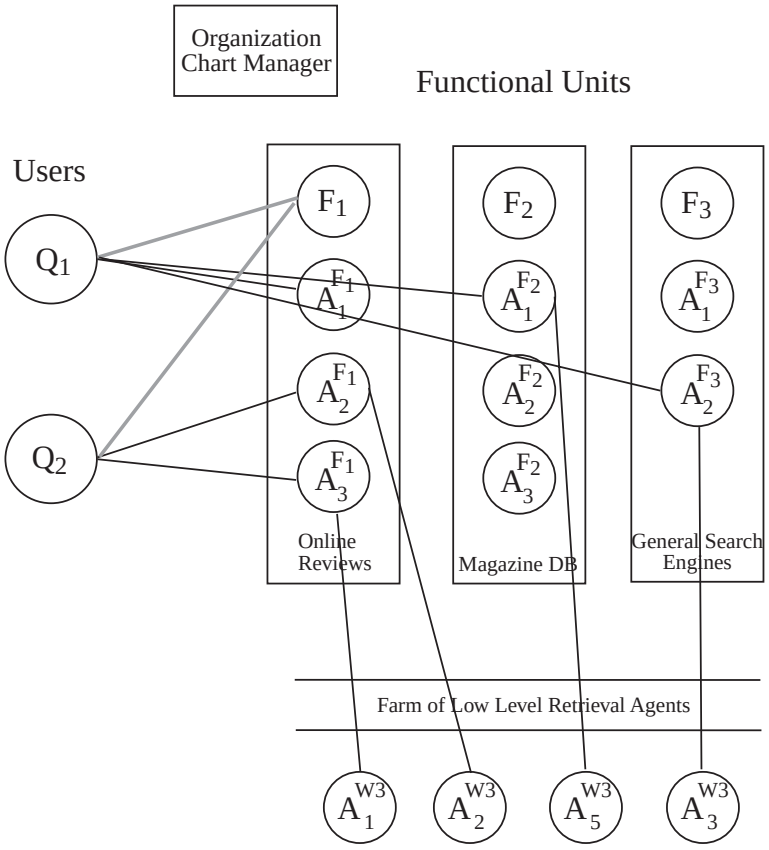

Figure 1: The MACRON Organizational Architecture

For example, in Figure 2 the query agent can expand the initial "Get Review" goal to "Get Online Reviews" and "Get Published Reviews." It then recognizes that it has to communicate with FM agents for "Online Reviews" and "Published Reviews." These managers in turn allocate the goals "Get Online Reviews" and "Get Published Reviews" to the agents in their respective functional units. Say in this example, each goal is allocated to one agent. The agents expand their goals to reach the lowest level primitive tasks like Use Wais, Use FTP, Retrieve Online Magazine(x) etc. The organization chart manager (OCM)

is an organizational memory store that allows an agent to get information about which agents to communicate with when shipping a task or sending coordination information. New agents and new functional units can be added to a running system by adding them to the organizational chart manager. For example, the query agent checks with OCM to find the address of the functional unit manager that can achieve "Get Online Reviews."

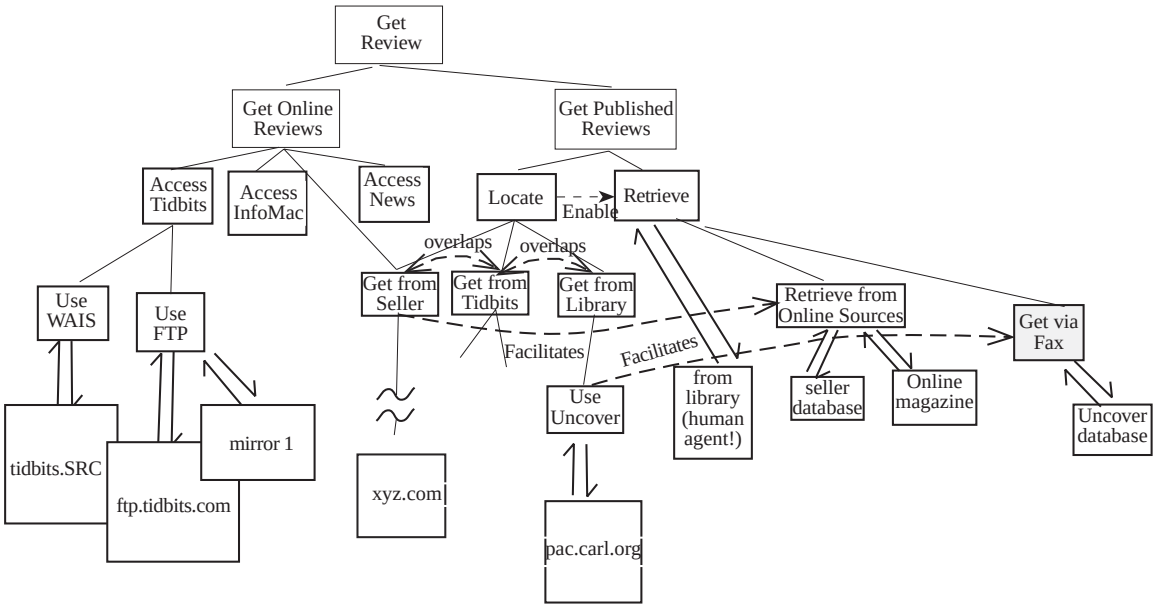

Figure 2: A goal tree for retrieving Macintosh related product reviews

Let us briefly compare the MACRON organizational approach to two alternate approaches. In a purely functional organization approach we would have fixed agent teams, each team consisting of at least one member from each functional unit. Such fixed teams cause several problems: since all queries won't need all functional resources, some team members will be idle. Conversely, if a necessary team member is temporarily unavailable, a query could fail. Third, as we introduce learning components into the agents, the teams begin to differentiate and develop different expertise in each of their functional areas—if the environment is dynamic, we can expect that new queries will appear that would be best answered by combinations of experienced agents that differ from the fixed teams offered. If we turn to the other end of the spectrum, with an undifferentiated sea of agents (who perhaps advertise with a central facilitator), a query handler can gather a team together to handle a particular query; there are no wasted resources, and no problems if a single agent is temporarily unavailable (such a sea of agents is more flexible than the fixed functional organization). However, this approach fails to provide simple methods for putting the best possible team together (especially in a cooperative system, where complex mechanisms for dealing with intentional cheating are not required). Which of the agents that advertise a service should the query agent actually contact and use? As agents learn and become differentiated, how can they compare their relative abilities without knowing about each other? In the MACRON organization choosing a functional agent is the sole responsibility of the functional managers (who act as specialized, intelligent facilitators). MACRON allows us the best of both worlds—the flexibility to put together an efficient team on the fly, and the

ability to intelligently and dynamically assign and track these agent resources.

2.2 The DECAF Agent Architecture

This architecture is based on the one used to test the GPGP (Generalized Partial Global Planning) coordination approach described in [4, 5]. In the context of Figure 2, for example, upon receiving its task "Get Online Reviews," the "Online Reviews" agent calls a planner to expand the task into a query plan consisting of primitive actions like "Use FTP" and "Use Uncover." A real-time scheduler is invoked in conjunction with coordination mechanisms to enable the agent to form a schedule that best serves the interest of the global query plan. For example, the scheduler realizes that "Use Uncover" facilitates the "Get via Fax" task in the "Published Reviews" agent and schedules it in the earliest possible slot to be able to communicate relevant partial results to the other agent. The execution monitor keeps track of the progress of plan execution and initiates appropriate measures for re-planning or rescheduling as needed.

Figure 3 shows the architecture of an individual agent consisting of several components:

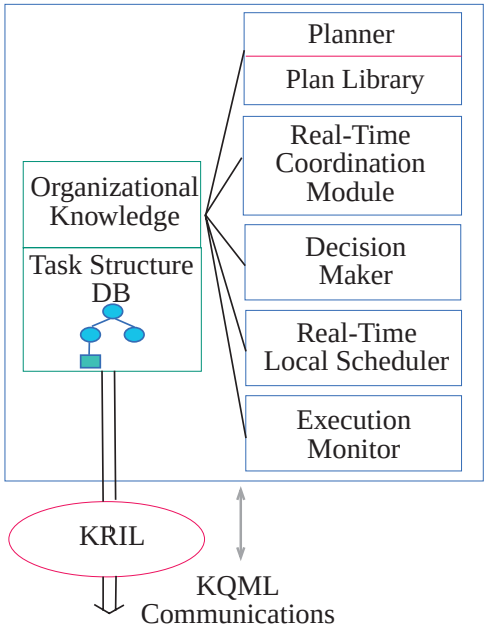

Figure 3: DECAF: Distributed Environment Centered Agent Framework

- A task structure database that contains the agent's current view of its tasks and their relation to other agent's tasks. This portion of the agent is a shared structure that is understood by all of the other active agent components. The planner creates new task structures here, the coordination module recognizes patterns here and creates new tasks or constraints, and the scheduler chooses, orders, and locates actions (leaf nodes of the task structure hierarchy) in time.

- A planning module that elaborates the task structure representation. The planner instantiates a task structure, based on the input query or input subtask, and the plan

library database. It binds the attributes of a task to specific values derived from the input query. The resulting task structure instantiation is placed in the agent's shared task structure database for use by the coordination and scheduling modules. Functional unit agents will have plan libraries that are specialized to their functions. Planning module capabilities can range from simple plan library retrieval to traditional classical hierarchical planning.

- A coordination module that recognizes certain patterns in the current task structures and responds by suggesting new meta-level tasks (such as communication with other agents) and possible scheduling constraints (such as local commitments to do a certain action before a certain time). The coordination module comprises several different mechanisms to be discussed below.

- A local scheduler that takes as input the current task structure, local commitments, and non-local commitments (from other agents) and produces several possible schedules (what actions to take, in what order, and at what point in time). A TAEMS task structure combined with some performance metric creates a well-defined NP-hard scheduling problem that we attack using a "Design-To-Time" heuristic scheduler [8]. The Design-To-Time (DTT) approach assumes that our task structures (retrieved by the planner) have multiple ways available to achieve tasks that trade-off the various performance criteria such as execution time and result quality. The DTT local scheduler then puts together (designs) a schedule that attempts to maximize performance criteria while meeting the deadline constraints. The DTT local scheduler that we use can actually return several schedules that each attempt to maximize different performance criteria (such as termination time, overall result quality, and fewest number of broken commitments). The DTT approach uses statistical information about average method execution times and does not schedule to worst case execution times, and so requires an execution monitoring component (below) to oversee the execution of individual actions. The DTT approach subsumes other popular real-time approaches such as anytime algorithms and imprecise computation[6]

- A decision-maker that chooses an appropriate schedule, based on the agent's current, dynamically changing view of its performance criteria. In general, the problems that DECAF agents are trying to solve have multiple performance criteria. For example, the the CIG domain, we need to worry about speed, the completeness of the solution (in the sense that all relevant information is retrieved), the certainty of the solution (in the sense that each solution facet is properly explored to some depth), network resource usage, and monetary costs (some information may cost money to access). Ignoring for the moment the complications of a multi-agent system, any particular single agent schedule of actions will necessarily trade-off these various performance criteria. The fastest schedule will not necessarily be the cheapest or the most complete, for example. We believe that in real-world problem-solving situations these criteria will need to change while the agent is working—either because different queries have different performance criteria or because that criteria changes during problem-solving. Rather than burden the already difficult local scheduler with this decision-theoretic balancing act, we create a decision-making component to handle it and allow the

scheduler to return multiple schedules that attempt to maximize different criteria (or, the use of several independent schedulers)[7].

- An execution monitor that oversees the executions of actions from the current schedule. The execution monitor is needed to anticipate problems and slipped deadlines. Given information about the statistical execution characteristics of task, the scheduler will schedule various monitoring points, and indicate the envelope that dictates when a method execution has fallen so far behind that the agent will need to reschedule to recover.

The CIG model[13] views information gathering as a distributed problem solving activity characterized by the existence of interdependencies between subproblems, leading to a need for the agents to coordinate extensively during problem solving. Lesser [12] presents the functionally accurate, cooperative (FA/C) paradigm as an approach to distributed problem solving. In FA/C systems, various soft and hard constraining *goal/task interrelationships* among subproblems motivate agents to coordinate their problem solving activities in order to enhance the efficiency of the ongoing problem solving process. The coordination module in a DECAF agent detects potential sources of missing scheduling information, subproblem interactions, and task redundancy. It then causes the creation of new local scheduling constraints, called commitments, which can be communicated to other agents. Various kinds of subproblem interaction or goal interrelationships like facilitates, enables, overlaps, and subsumes that exist between subproblems can be detected and exploited in a variety of ways [4, 5]. Different coordination mechanisms are used to deal with different classes of interdependencies. Some example mechanisms we have developed include:

- a mechanism that recognizes and responds to tasks that have an impact on other agents but are not believed to be known by the other agents
- a mechanism that communicates task results when they will help another agent (but are not necessary)
- mechanisms that deal with coordinating schedules around redundant activities, or hard- or soft-coordination relationships.
- mechanisms that hand off tasks to other agents when they fit the other agent's organizational role (one can view a role as a long term commitment to a certain class of actions).

In our example in Figure 2, locating a paper review "enables" its retrieval, i.e. paper reviews may be obtained by first finding a citation, and then either finding the actual article or obtaining it from the publisher. Finding a citation via Uncover "facilitates" the goal of getting the article faxed to the user. An overlaps interrelationship exists between goal "Get from Seller" and goal "Use Uncover." Once the seller's archive offers a particular citation, the search for that same citation at the Uncover database can be avoided. Another source of information that is not exploited in the example above is the increasingly popular World Wide Web (WWW). We can think of the web of citations or the web of hyper-text links associated with a document as a web of consistency constraints. That is, documents

linked in this way may contain related information. For example, during the process of retrieval of product review discussed above via the WWW, two sites may quote different prices for the product. Product review information at an FTP site accessed via the WWW may contain outdated prices, whereas a link to the seller database in a html document may in fact contain the latest prices. When inconsistencies are uncovered, agents need to work to resolve the associated uncertainty so that a cluster of consistent documents containing "correct" information is presented to the user. In this case, the agents may choose to use the seller database information and override other sources for the price information.

2.3 Low-level Retrieval Agents

The low-level retrieval agents are the system's interface to the World-Wide-Web[3]. The agents, referred to as "getters," are responsible for retrieving and processing URLs for the higher-level planning agents and interacting with remote search engines and databases. Each of the primitive tasks in a schedule developed by a higher-level agent results in the recruitment of one or more of these low-level retrieval agents to achieve the task. Getter services include MIME type retrieval via URL specification, HTML parsing, keyword match and selection of HTML or text documents, selection and extraction of embedded URLs, and the ability to interact with HTML forms. Like the other agents in the MACRON architecture, getters communicate using KQML. Getters use domain specific knowledge about particular sites and fill-in HTML forms to determine how to interact with remote search engines and query forms and how to properly process their output. Through these services the getters provide a necessary abstraction of the WWW and its heterogeneous information resources to the rest of the agent system.

2.4 User Interface

In developing the user interface, we vested the human users with the ultimate authority to override or redirect the search process as they deem appropriate. However, such a philosophy gives rise to the necessity for careful design of the user interface. The time-line of a complex multi-agent search process has to be displayed to the user in a cognitively compatible manner, and his or her feedback has to be mapped back to the distributed search control process.

The user interface for MACRON displays the evolving task structures, and the user can choose a task to get a more detailed view of its status. The user is permitted to make changes to the modifiable parameters of the task and submit the changes which are then mapped back to the agents to be assimilated into their search control.

3 Status of Implementation

MACRON is in its early stages of development. The basic agent communication infrastructure is the KQML-based Lockheed KAPI integrated with Harlequin Lispworks. We have a rudimentary multi-agent information gathering system[1] with a Mosaic forms interface that receives a query from the user and invokes a query agent. The query agent communicates

[3]Our agents are not necessarily tied to the WWW and can access any general source of information

with a hierarchical case-based planner that instantiates an information gathering plan and delegates primitive retrieval tasks to the appropriate low level agents. These agents in turn communicate with the actual information sources. As of now we have low level agents that can communicate with several different information sources (Djanews, Macinfo, INQUERY etc.) available on the World Wide Web. The planner agent is set up to answer queries on Macintosh products.

4 Conclusion

Information Gathering systems proposed in the literature typically either do not fully exploit the potential of knowledge-intensive methods for the task [2, 3, 9] or fail to exploit the dependencies between agents working on different aspects of an information gathering task[14]. The system by Knoblock et al.[10] is an exception in the sense that it is knowledge-intensive and has some notion of exploiting dependencies. However, their system has no notion of scheduling to maximize quality given deadlines and cost limits. MACRON is an attempt to overcome these shortcomings. MACRON is based on our strong belief that cooperating to enhance efficiency of a resource-limited information acquisition process in complex, heterogeneous and unstructured data environments can be viewed as a distributed problem-solving task within the FA/C paradigm.

References

[1] CIG Searchbots
. `http://dis.cs.umass.edu/research/searchbots.html`.

[2] M. C. Bowman, P. B. Danzig, U. Manber, and M. F. Schwartz. Scalable internet resource discovery: Research problems and approaches. *Communications of the ACM*, 1994.

[3] J. Callan. Personal communication, July 1994.

[4] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, jun 1992.

[5] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI Press. Longer version available as UMass CS-TR 94–14.

[6] Alan Garvey. *Design-to-time Real-time Scheduling*. PhD thesis, University of Massachusetts, 1995.

[7] Alan Garvey, Keith Decker, and Victor Lesser. A negotiation-based interface between a real-time scheduler and a decision-maker. CS Technical Report 94–08, University of Massachusetts, 1994.

[8] Alan Garvey and Victor Lesser. Design-to-time scheduling with uncertainty. CS Technical Report 95–03, University of Massachusetts, 1995.

[9] M. Huhns, U. Mukhopadhyay, L. M. Stephens, and R. Bonnell. DAI for document retrieval: The MINDS project. In *Distributed Artificial Intelligence.*

[10] C. A. Knoblock and Y. Arens. An architecture for information retrieval agents. In *Working Notes of the AAAI Spring Symposium on Software Agents.*

[11] C. A. Knoblock, Y. Arens, and C. Hsu. Cooperating agents for information retrieval. In *Proceedings of the 2nd International Conference on Cooperating Information Systems*, 1994.

[12] V. R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347–1363, November 1991.

[13] T. Oates, M. V. Nagendra Prasad, and V. R. Lesser. Cooperative Information Gathering: A Distributed Problem Solving Approach. Computer Science Technical Report 94–66, University of Massachusetts, 1994.

[14] E. M. Vorhees. Software agents for information retrieval. In *Working Notes of the AAAI Spring Symposium on Software Agents.*