MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 204 September 1980

REPORT ON THE WORKSHOP ON DISTRIBUTED AI

Randall Davis

Abstract

On June 9-11, 22 people gathered at Endicott House for the first workshop on the newly emerging topic of Distributed AI. They came with a wide range views on the topic, and indeed a wide range of views of what precisely the topic was.

In keeping with the spirit of the workshop, this report describing it was prepared in a distributed fashion: Each of the speakers contributed a summary of his comments. Sessions during the workshop included both descriptions of work done or in progress, and group discussions focused on a range of topics. The report reflects that organization, with nine short articles describing the research efforts, and four summarizing the informal comments used as the foci for the group discussions.

Introduction

On June 9-11, 22 people gathered at Endicott House for the first workshop on the newly emerging topic of Distributed AI. They came with a wide range views on the topic, and indeed a wide range of views of what precisely the topic was.

The meeting as organized with several purposes in mind. One motivation was the issue of definition, as we attempted to establish an informal consensus on the meaning and scope the term. The workshop also served to bring together advocates of a number of different approaches to the problem, making possible useful comparisons and heated exchanges. The meeting provided as well a chance to identify the relatively small and widely dispersed community of people working on the topic.

What is DAI?

Despite the broad range of perspectives presented, a number of common themes emerged. Distributed AI is concerned with those problems for which a single problem solver, single machine, or single locus of computation seems inappropriate. Instead we turn to the use of multiple, distinct problem solvers, each embodied in its own system.

Use of this approach may be motivated by attributes of the problem itself or by the desire to focus on research issues that are best studied in a distributed environment. In the most obvious case of the former, for example, the presence of multiple agents (e.g., multiple robots) leads us to think in terms of multiple, cooperating entities. The availability of a natural decomposition also suggests this approach. In a sensor net, for example, an obvious (geographic) decomposition is implied by the sub-area covered by each sensor. In other problems a clear functional decomposition may be possible, and it may be useful to embody each function in a distinct machine. Data rates may also be a contributing factor, as in the sensor net, where centralized processing is impossible due to the volume of data to be processed. The presence of any of these factors suggests that the problem may be well suited to being attacked with a group of distinct, cooperating problem solvers, rather than the single, centralized system traditionally used in AI.

As some of the accompanying articles make clear, the DAI approach may also be motivated by research interests. There is reason to believe, for instance, that many issues in problem solving and reasoning about knowledge and action are brought into sharp contrast when put into this setting. There is also interest in the approach as a means of building more powerful problem solvers: It may prove much easier to coordinate the activities of twenty medium sized machines than it is to build a single machine twenty times (or even ten times) as large. It may even be argued (as Nilsson does) that DAI may be a prerequisite to ordinary AI.

Less widely agreed upon, but still important themes in much of the work reported, are the concepts of problem solving in the context of incomplete or inconsistent information, the difficulties inherent in distributed control, and the significance of limited bandwidth.

If, for example, we have a large problem spread across multiple distinct machines, then we may have a situation in which each machine can know about only part of the problem. How can we ensure effective problem solving in the face of incomplete information? With a large collection of machines it may require unreasonable effort to ensure that all data bases are always consistent. Are there approaches to problem solving that are relatively less sensitive to such inconsistencies?

The need to distribute data has long been recognized, but we want to distribute control as well. Given a collection of relatively autonomous problem solvers, how do we ensure coherent behavior? How can we prevent the systems from working at cross purposes?

With current or even foreseeable technology, it is much cheaper to compute than it is to communicate; bandwidth thus becomes the limiting resource. Can we design approaches to problem solving that ensure loose coupling between processors, so that the machines can spend most of their time computing, and much less of their time communicating?

...as opposed to distributed processing

This work differs in several respects from the more widely known topic of *distributed processing*. Typically, for instance, most of the processing in such systems is done at a central site and remote processors are limited to basic data collection. Control is not distributed and the processors do not cooperate in a substantive manner.

A second distinction arises from examining the origin of the system and the motivations for interconnecting machines. Distributed processing systems often have their origin in the attempt to synthesize a network of machines capable of carrying out a number of widely disparate tasks. This often leads to a concern with issues such as access control and protection, and results in viewing cooperation as a form of *compromise* between potentially conflicting views and desires at the level of system design and configuration.

In distributed problem solving, on the other hand, there is a single task envisioned for the system; the resources to be applied have no other predefined roles to carry out. This means that we view cooperation in terms of benevolent problem solving behavior, i.e., how can systems that are perfectly willing to accommodate one another act so as to be an effective team. Our concerns are thus with developing frameworks for *cooperative behavior between willing entities*, rather than frameworks for enforcing cooperation as a form of compromise between potentially incompatible entities.

This Report

In keeping with the spirit of the workshop, this report was prepared in a distributed fashion: Each of the speakers contributed a summary of his comments. Sessions during the workshop included both descriptions of work done or in progress, and group discussions focused on a range of topics. This report reflects that organization: nine articles describe the research efforts, the other four summarize the informal comments used as the foci for the group discussions. Because it argues the case so well, we start with Nils Nilsson's informal commentary on basic motivations for doing DAI. This is followed by the nine research descriptions, then the three remaining discussion session summaries.

TWO HEADS ARE BETTER THAN ONE

Nils J. Nilsson
SRI International

Those of us doing research in distributed artificial intelligence are often asked why are we trying to solve the problem of building multiple, cooperating artificial intelligences when we haven't yet solved the problem of building just one of them. There are several reasons why I think that work on DAI will contribute to (and may even be a prerequisite for) progress in ordinary artificial intelligence.

1. To be sufficiently "intelligent," a system may have to be so complex and may have to contain so much knowledge that it will be able to function efficiently only if it is partitioned into many loosely coupled subsystems. Hewitt's and Kornfeld's "scientific community" metaphor, Minsky's and Papert's "society of minds," and (to some degree) "frame-based" systems all proclaim "No AI without DAI."

2. Work in DAI helps sharpen our intuitions and techniques for explicit reasoning about knowledge, actions, deduction and planning. In my opinion, we have yet to devise entirely satisfactory methods for representing beliefs, plans, and actions so that these concepts can be reasoned about. The objective clarity gained by considering how one AI system can reason about another should illuminate our study of how an AI system can reason about itself.

3. Methods used by one AI system for reasoning about the actions of other AI systems will also be useful for reasoning about other dynamic (but unintelligent) processes in the environment. (Note that such AI systems might occasionally make the mistake of attributing planned behavior to purpose-less processes and thus might develop animistic theories about its environment.) Previous work in AI planning methods largely dealt with only static environments.

4. DAI work will advance our techniques for automatic planning systems in the following ways:

   a. **Hierarchical Planning:** Techniques that enable a system to incorporate the effects of the actions of other systems in its plan can also be used to incorporate the effects of low-level actions in higher levels of the plan. (Reasoning about the plans of others is similar to reasoning about the details of our own plans.)

   b. **Information Gathering:** Techniques that enable a system to plan to ask another system for information can also be used to plan to acquire information through sensors.

   c. **Execution Monitoring:** Techniques that enable a system to check the performance of other systems can be used in monitoring the execution of its own plans.

5. DAI work will contribute to our understanding of the process of natural language communication. The "communication acts" between intelligent systems serve as an abstract model of some aspects of natural language generation and understanding. Viewing the process abstractly may clarify certain problems in natural language communication.

6. An AI system that can reason about other AI systems can also reason about its human user to maximize its utility to that user.

**** PROJECT REPORTS ****

OVERVIEW OF MULTIPLE AGENT PLANNING SYSTEMS

Kurt Konolige and Nils Nilsson
SRI International

Certain tasks can be more advantageously performed by a system composed of several "loosely coupled," cooperating AI agents than by a single, tightly-integrated system. These multiple agents might be distributed in space to match the distributed nature of the task. Such systems are often called distributed AI (DAI) systems. We are interested here in systems in which the component agents themselves are rather complex AI systems that can generate and execute plans, make inferences, and communicate with each other.

Among the potential advantages of such DAI systems are graceful (fail-soft) degradation characteristics (no single agent need be indispensable), upward extensibility (new agents can be added without requiring major system redesign), and communication efficiency (a message-sending agent can plan its "communication acts" carefully taking into account the planning and inference abilities of the receiving agents).

In planning its actions, each agent must consider the potential actions of the other agents. Previous AI research on systems for generating and executing plans of actions assumed a single planning agent operating in a world that was static except for the effects of the actions of the planning agent itself (STRIPS, NOAH, NONLIN, etc.). Several important extensions must be made to planning systems like these if they are to function appropriately in an environment populated by other planning/execution systems.

First, each agent must be able to represent certain features of the other agents as well as the usual information about static objects in the world. Each agent must have a representation for what the other agents "believe" about themselves, the world, and other agents. Each agent must have a representation for the planning, plan-execution, and reasoning abilities of the other agents. These requirements presuppose techniques for representing the "propositional attitudes" believe and want. Second, among the actions of each agent are "communication actions" that are used to inform other agents about beliefs and goals and to request information. Finally, each agent must be able to generate plans in a world in which actions not planned by that agent spontaneously occur. We introduce here the notion of spontaneous operators to model such actions.

We have developed a formalism for agents, taking the above ideas into account. The formalism is based on Wehyrauch's notion of multiple first-order theories and metatheories, extended to incorporate goals and planning procedures for each agent. We have experimented on paper with several small blocks-world examples in which agents must take other agent's planning activities into account when forming their own plans. Currently, we are exploring deduction and control strategy issues within the formalism that are necessary to reason efficiently in such a framework, with a view towards implementation. Finally, while we believe that the framework we propose is general enough to deal with widely varying domains where agents must reason about other agents' plans (e.g., helpful assistants, sensor nets), it will be necessary to pick a particular domain within which to exercise the framework.

Selected References

Konolige K, Nilsson N, Multiple agent planning systems, SRI International.
Appelt D, A planner for reasoning about knowledge and belief, SRI International.

DISTRIBUTED ARTIFICIAL INTELLIGENCE AT CARNEGIE-MELLON UNIVERSITY

Report by Mark Fox

Distributed Sensor Nets

Researchers involved in DSN:

Richard F. Rashid	Rick Snodgrass
Peter Hibbard	David Hornig
Horst Mauersberg	Robert Fitzgerald
Raj Reddy	Bernd Bruegge
Hank Mashburn	

The CMU distributed sensor net (DSN) is a testbed project designed to explore the issues and problems of distributed signal understanding. The basic system will locate, identify and track a single remotely controlled model vehicle in a simulated terrestrial environment monitored by an array of acoustic sensors. The long-term goal of the system is to have it serve as an intelligent assistant which can make decisions in the presence of uncertainty, conflict, delay and lack of a priori hypotheses.

The DSN system will consist of a set of cooperating processes on a number of host computers connected via a high-speed network. An important part of the project is the design and construction of a programming environment for distributed systems, including support for message-based communication between processes, symbolic task-to-processor mapping, and various debugging aids. Initial work in this area has taken the form of the design and implementation of an inter-process communication facility for VAX/UNIX [Rashid 80].

Fault tolerance is an explicit goal. The system should be able to trade performance for functionality and should operate independent of the actual network, host, or microphone configuration (within the limits of the acoustic algorithms being employed). There will be an explicit division of the system into components which make policy and components which provide application-specific functions. We intend to incorporate knowledge about a the dynamic state of the system (e.g. location of sensors, topology of network, b) the state of inter-process communication), and knowledge of the task being performed (in terms of a user-supplied description of the communication requirements and operational constraints of processes). By using this information, the system will be able to map the object location algorithm onto the available processors and recover from failures of individual processes or whole processors.

The current state of the project (6/26/80) is as follows: The basic algorithms for object location have been designed, implemented, and tested on synthetic data. The inter-process communication (IPC) facilities have been implemented and are operational under VAX/UNIX. The IPC network protocol design (based around Internet datagrams) is nearly complete. Work is proceeding on the design and implementation of a language for the description of the static and dynamic interconnections required by DSN processes. Work is also proceeding on the design and implementation of a multi-process debugger.

ORGANIZATION STRUCTURING

Mark Fox

A study was made of factors affecting the structure of large, complex systems. Human organizations were surveyed for solutions to organizational problems. Consequently, a report was written surveying a number of fields to ferret out useful design (structuring) concepts. Control structure research in Artificial Intelligence was briefly surveyed. Organization Theory, a field in management science, provides interesting approaches based on analyses of complexity, uncertainty and behavior. Economic Team Decision Theory provides an analytical approach to measuring alternative (distributed) organizations of (rule-based) programs. Finally, a language was defined incorporating many of the concepts elicited from the surveyed fields.

INTELLIGENT INFORMATION SYSTEMS

Mark Fox, Don Cohen, Ramona Reddy, Tom Morton

A distributed system is currently being designed and constructed to manage and control a factory environment. The factory characteristics are: a large number of different but similar products are produced, a product can be produced in more than one way, a large number of machines are available, there is high resource contention (e.g., tools, materials, people, etc.), and orders of varying size and priority enter the factory. The first version of the system will have distributed sensors with centralized control. The second version will distribute management and control to processors spread throughout the factory. Processors will have overlapping areas of sensing, cooperate in determining plans and schedules, and propagate information to responsible processes. Processor failure will be handled by task sharing and migration, and redundancy in information monitoring and storage.

The current state of the system is: An AI knowledge representation system has been constructed as a basis for the factory model and database [Fox, 1980]. Models for two separate factories are under construction and simulation system has been constructed that runs off the model. A color graphics display system of the state is under construction, and work is proceeding on factory scheduling algorithms.

Selected References

Fox M.S., Organization structuring: designing large, complex software, Technical Report, Computer Science Department, CMU, 1979.

R. Rashid, An inter-process communication facility for unix, Technical Report, Computer Science Department, CMU, 1980.

Fox M.S. Structuring: knowledge accommodation and discovery using specialization and planning, (Ph.D. Thesis), Computer Science Department, CMU, to appear, 1980.

COOPERATIVE PROBLEM SOLVING WITH THE CONTRACT NET

Randy Davis (MIT)
Reid Smith (DREA, Canada)

Work on the contract net has focused on developing protocols for interaction between independent, cooperating problem solvers. The fundamental idea is to structure interaction as a process of **negotiation**. To illustrate a simple example of this process, consider a collection of processor nodes trying to work together to solve a problem. Initially one node is given the entire problem description. If the problem turns out to be too big to be solved by that node alone, the node partitions the problem into smaller pieces. It keeps one of those pieces to work on and solicits help on the others. It does this by issuing *task announcements*, broadcast messages that describe the tasks to be done.

Idle nodes listen to these announcements and evaluate their relative interest in and ability to perform each task. After listening to several announcements, an idle node may decide to submit a *bid* on one of the tasks, a message that includes information indicating why that node may be especially able to perform the task in question. The node that announced the task may receive several bids and can choose the one that seems most appropriate. It sends an *award* message to inform the successful bidder that it is now responsible for the task. We call the successful bidder the *contractor* and refer to the node that announced the task as its *manager*.

The process can then recur, with the contractor further partitioning the task, announcing sub-tasks, listening to bids, and awarding sub-tasks. It then becomes the manager for the tasks that it announced. When a node finishes a task, it reports the result back to its manager with an *award message*.

Note that tasks are distributed via a discussion that involves a simple form of negotiation: Idle nodes evaluate a number of task announcements and choose the one they prefer. Managers then evaluate a number of bids and choose the one they prefer. The resulting "agreement" is thus a process of local, mutual selection.

In our view, distributed problem solving in the most general sense can usefully be divided into four phases: (i) problem decomposition, (ii) sub-problem distribution, (iii) sub-problem solution, and (iv) answer synthesis. Three of these are the well-known concepts from standard (i.e., centralized) AI problem solving, the second is introduced by the new environment. In our experience, however, even the standard notions change in a distributed setting due to the availability of multiple processors and the limitation on communication bandwidth. For example, we want the system to be able to reason about the appropriate problem decomposition to make (phase i), to ensure a good decomposition despite variations in available resources. We also want to supply a mechanism that permits careful matching of problems to available processors (phase ii), to ensure appropriate use of resources. Finally, we need a mechanism that permits efficient communication between processors working on related tasks, to effect cooperative problem solving during phase iii.

At the workshop we reported on preliminary efforts to attack these issues, using the Waltz problem of understanding line drawings of blocks. Previous applications of the contract net (a simple model of a distributed sensor net [Davis, 80] and heuristic search [Smith, 80]) have relied in large part on the traditional hierarchical decomposition and independent solutions of subproblems. The Waltz

problem was chosen because it is well suited to a cooperative style of solution.

The problem per se has of course long been solved for the serial case (the original implementation); the extension to the parallel case is clear (put a processor at each vertex). But truly *distributing* the solution effort among a collection of distinct problem solvers requires mechanisms for decomposition and distribution, as well as cooperative problem solving. The problem thus offers a useful basis for studying a range of issues.

To get started, we need a statement of the problem. One plausible description of a scene would be a list of vertices and their interconnections. This list would be handed to one of the processors. Problem decomposition involves dividing up the set of vertices; distribution requires a mechanism to hand out the resulting subtasks. Ideally, both decomposition and distribution should be insensitive to the number of processors available (to ensure flexibility and the ability to redistribute in case of failure or overload).

We discussed the exploration of a modified minimum cut set algorithm. This makes possible a decomposition and distribution which is itself distributed (in the sense that no one processor is handing out all the tasks), produces a good decomposition independent of the number of processors, and reduces communication overhead (by keeping related subproblems in the same node as long as possible).

Cooperative problem solving behavior requires some way of establishing communication between nodes with related subproblems. This is sometimes apparent in the task itself (in a DSN, for example, it is typically true that physically adjacent nodes should communicate). In general, however, we need some mechanism for establishing the appropriate lines of communication. We reported on a simple labeling mechanism which fits easily into the contract net protocol, and allows nodes with interacting subproblems to identify one another and hence communicate directly. These nodes then exchange messages describing their current set of vertex labels, as in the original solution.

This work is the first step in a series of experiments currently under way to explore the issues of problem decomposition and distribution, cooperative behavior, and answer reconstruction in distributed problem solving.

Selected References

Davis R, Smith R G, Negotiation as a metaphor for distributed problem solving, in preparation.

R. G. Smith, A framework for problem solving in a distributed processing environment, thesis, STAN-CS-78-700 (HPP-78-28) Dept. of Computer Science, Stanford University, December 1978.

Smith R G, Applications of the contract net: search, *Proc 3rd CSCSI Conference*, May 1980.

Smith R G, Davis R, Frameworks for cooperation in distributed problem solving, to appear in *IEEE SMC*.

THE USC/ISI DSN PROJECT

Jeff Barnett
USC/ISI

Distributed sensor networks (DSN) systems are real time, loosely coupled, cooperating systems. They combine information from many sensors to produce a unified world picture. Their most interesting features are: (1) the entire system, including the application processes, communication mechanism, and operating systems, share a common goal. Therefore, resource allocation is not competitive in the traditional sense, but rather must be for the good of the whole system; and (2) knowledge about priorities is distributed both physically and vertically throughout the entire system. Both points are in sharp contrast to, say, the ARPA network community.

The DSN project at USC/ISI has selected a simple air defense game as its research context. The initial version of the game is played by a single user who receives (simulated) sensor data input and directs the system's response. The single system objective in the game is to minimize loss due to enemy attack by controlling defensive weapons. As portions of the user's behavior are understood, they are automated. Since the node supporting the user is resource limited, some of these functions must be distributed. Thus, the game provides not only a context for evaluating technology, but a top-down forcing function to evolve a distributed system.

The game is played in a testbed. The testbed provides simulation facilities for such things as sensors, computation facilities, and communications. Also provided is a program development environment for application level processes.

The DSN project is investigating several areas of general import to computer science and AI. Research ideas will be evaluated in the context of the game. That is, does a proposed methodology improve one's ability to play, and what does it cost? The research areas are:

Interface: What information and controls does the user need in order to accomplish his task, how is it presented, and how does the system explain and defend automated portions of its behavior?

Resources: How does application level knowledge effect resource allocation by the system level components, particularly the communications mechanism?

Inference: How is errorful information with diverse representations and underlying models combined? The fusion problem.

Organization: How are responsibilities bound to processes and processes bound to processors, both initially and after configuration change or compromise.

Representation: How and should a distributed system maintain a global and consistent representation of objects?

Computation: What are theoretical models of distributed computation? Characterize communication and decision making procedures.

Data management: What are the tradeoffs over data locality, redundancy, and consistency requirements?

Selected References

Barnett J, DSN - Distributed sensor networks: working papers, ISI/WP-12, USC/ISI, April 1979.

COOPERATIVE DISTRIBUTED PROBLEM-SOLVING AND ORGANIZATIONAL SELF-DESIGN

Victor Lesser and Dan Corkill
UMass at Amherst

The major focus of our current research efforts has been the exploration of a new model for distributed problem solving systems. This model differs from conventional approaches in its emphasis on dealing with distribution-caused uncertainty and errors in control, data, and algorithm as an integral part of the problem solving process. Much of the inspiration for this new model comes from the mechanisms used in knowledge-based Artificial Intelligence (AI) systems to resolve uncertainty caused by noisy input data and approximate knowledge. This approach is especially suited to applications in which the data necessary to achieve a solution cannot be partitioned in such a way that a node can complete a task without seeing the intermediate state of task processing at other nodes.

In this model, nodes cooperatively problem-solve by exchanging partial, tentative results (at various levels of abstraction) within the context of common goals. In a distributed system based on this model, it is often appropriate to have node activity be *self-directed*. For instance, if a node does not receive an appropriate partial result in a given amount of time, it is able to continue processing, utilizing whatever data are available at that time. From this self-directed perspective, it is more appropriate to view a distributed system as a complex organizational structure which is *synthesized* from local systems operating at each node.

In our experiments with such systems, we have come to realize that one of the key issues in effectively exploiting such an approach is how to maintain global coherence among the autonomous, self-directed problem-solvers. This issue has led us to realize the need for organizational self-design. Organizational self-design is the explicit planning by the system of the pattern of information and control relationships that exist between the nodes in the system and the distribution of problem solving capabilities (expertise) among the nodes in the system. As the problem solving environment changes, the distributed system may need to change its organizational structure to maintain its effectiveness. In order to effect such a change, the distributed system must (1) detect the decreased effectiveness of its organizational structure; (2) propose alternative organizational structures; (3) evaluate the cost of continuing with its current organizational structure versus reorganizing itself into a more appropriate structure; (4) carry out such reorganization if appropriate.

We are currently developing a distributed interpretation testbed. The testbed simulates a model of a distributed, knowledge-based, problem-solving architecture applied to an abstracted version of a vehicle monitoring task. This testbed will permit us to empirically explore a wide range of control/communication policies for organizing distributed problem-solving systems. In the context of this testbed, we are developing a distributed planning system which integrates both the coordination of "domain-level" activities and the construction and maintenance of an appropriate organizational structure.

Selected References

Corkill, D. D., An organizational approach to planning in distributed problem solving systems, TR 80-13, Department of Computer and Information Science, UMass. Amherst, May 1980.

Lesser, V.R., and D.D. Corkill, Functionally-accurate cooperative distributed systems. *Proceedings of the International Conference on Cybernetics and Society*, pp. 346-353, October 1979.

Lesser, V.R., and L.D. Erman, An experiment in distributed interpretation, *Proceedings of the First International Conference on Distributed Computing Systems*, pp. 553-571, October 1979.

DISTRIBUTED INTELLIGENCE FOR AIR FLEET CONTROL

Robert Wesson, Randy Steeb, and Frederick Hayes-Roth
The RAND Corporation

Coordination of air traffic is an area in which distributed artificial intelligence can make substantial contributions. This problem area is characterized by spatially distributed data, high communication costs, complex problem-solving activities, and high reliability requirements. Rand is engaged in a research program to establish methodologies through which individual cooperating processors will be able to sense situational conditions efficiently, exchange information, interact with human controllers, and coordinate and control system operations adaptively.

The focus of this program is the exploration and implementation of a number of architectures for distributed planning and control. Among the major architectures considered are 1) Space-centered, in which each processor has responsibility for aircraft within a geographic area, 2) Function-centered, in which each controller has responsibility for a different mission function, 3) Object-centered, in which each controller is assigned to a different aircraft, and 4) Plan-centered, in which different processors are assigned to different plans or sub-plans leading to attainment of an overall goal. These architectures differ in the way they decompose tasks and assign situation assessment, planning, and execution components to processors. At the same time, the architectures define the types of communication links, distribution of loads among processors, forms of planning, and flexibility of system response to changing task demands.

Experience with the architectures in a simulated air traffic control task has shown the importance of the form of planning employed. The possible distributed planning forms include 1) Pre-planned actions, 2) Individual, localized planning and replanning, 3) Asynchronous, cooperative planning with partial plan communication, and 4) Simultaneous, cooperative planning. These forms of planning differ strongly with respect to communication demands, processing requirements, and problem solving capabilities.

The goal of our work will be the development of a theory of architecture, encompassing the logical qualities of the structures (the theoretic performance independent of implementation), the engineering constraints induced by physical realization, and the implications of changing task environments. Through a series of analytic and experimental efforts, we hope to establish quantitative functional relationships between architecture structures, task characteristics and system performance measures.

Selected References

Steeb, R., B. Wesson, S. Cammarata, and F. Hayes-Roth, Distributed air-fleet control: planning and control architectures, The Rand Corporation, in progress.

Wesson, R., Air traffic control: whither goest thou?, The Rand Corporation, in progress.

Wesson, R., et al, Network structures for distributed situation assessment, R-2560-ARPA, The Rand Corp., August 1980.

MESSAGE PASSING SEMANTICS

Carl Hewitt and Bill Kornfeld
MIT

Hewitt and Kornfeld presented two projects that are part of MIT Message Passing Semantics Group's effort to produce highly parallel problem solving systems and hardware to run them on.

Hewitt first discussed the design of the *Apiary*, a machine architecture designed for running programs in actor based languages. The apiary consists of a large number of interconnected *workers*. Each worker consists of a processor, integrated circuit memory, and communications processor that allows it to communicate with other workers.

Workers are connected in a topology known as the *folded hypertorus*. To understand how a hypertorus is constructed, first imagine the processors arranged in a large cube. Each processor is connected to six others, its immediate neighbors in three directions. Now take corresponding processors on opposite faces and join them with long wires. This is the hypertorus. There is a way of folding the hypertorus in such a way that all wires become short (independent of the size of the net).

Jeff Schiller has constructed a preliminary implementation of the Apiary on the MIT Lisp Machine System using the Chaos Network. This simulation is being used to design and test load balancing and actor migration algorithms for the Apiary. A prototype communications chip has also been designed and implemented by Phyllis Koton.

Kornfeld discussed some work done with the design of parallel problem solving systems. He began by discussing the concept of *combinatorial implosion*. When a problem is to be solved, attempting multiple approaches simultaneously can be advantageous because the ultimately successful approach cannot be known in advance. Partial information can be used to reapportion the system resources in ways that minimize the overall expected running time. A language known as *Ether* has been designed and implemented on the MIT Lisp machine for running highly parallel programs.

A problem solving paradigm was explained in which several kinds of activities are run in parallel. When a problem is posed to the system, *proposers* propose possible solutions. *Proponents* attempt to show the solutions work and *skeptics* attempt to show they will not accomplish the goal. *Evaluators* examine partial results as they accrue and reapportion processing power to the various approaches in ways that fit current evidence.

One justification for this approach, known as the "Scientific Community Metaphor" was discussed. Scientific communities are systems that are capable of solving problems. Whenever there is an open question, different researchers will inevitably pursue different approaches to its solution. As new information is learned the total resources of the community are adjusted so that more successful approaches receive more support.

One important characteristic of the distributed AI work in the MIT Message Passing Semantics group is that details of the hardware, such as the number and configurations of physical processors, are hidden from the programmer. Programs are written using concepts that are convenient to the programmer and then lower level mechanisms, analogous to virtual memory, map these language constructs onto the physical hardware.

Selected References

Hewitt C, The APIARY network architecture for knowledgable systems, *Proc. LISP Conf*, 1980.

Kornfeld W, Using parallel processing for problem solving, MIT AI Memo 561.

SUMMARY OF ROCHESTER GROUP EFFORT

Dana Ballard
University of Rochester

Work at Rochester relevant to Distributed AI is centered on is brain modeling. The objective of this work is to develop abstract connectionist models of the brain that can explain sensory and motor processing. The current focus is threefold: (1) models of low and intermediate level vision; (2) abstract computational problems that arise from the connectionist model; and (3) models of motor activities.

At the DAI Conference the vision effort was emphasized. Recent research in vision has revealed the capability of distributed systems to compute intrinsic images of range, velocity, occluding contours, and surface orientation from the "primal" grey-level input. In each case this can be done by iteratively refining fuzzy, local parameter measurements.

To use these inputs, they must be organized in some way that relates them to objects. Our model of intermediate-level vision postulates the existence of many transformations that map portions of intrinsic images into useful features. Also important are feature-to-image transforms which map feature values onto intrinsic image regions. Such mappings can explain aspects of model-directed perception. The uniqueness of our approach is based on:

* the mathematical description of these transforms;

* a connectionist implementation of them.

Our generalization of the Hough transformation to detect shape from contours was used to illustrate these points.

The Rochester group, like most of the other groups at the conference, is taking a unique DAI approach, but there are three aspects of our work that are of special interest. The first is the connectionist model. In its purest form, processors are measurement values and can only communicate confidences about these values. The second is the totality of the problem distribution; the input, the algorithms, and the output are all distributed and are never centralized. This contrasted with problems being tackled by other groups which were distributed, solved in a distributed manner, and then reassembled. The third aspect is the resource-rich computing environment of the brain, which does away with the need for much time-sharing. Again in contrast, other groups are tackling areas where the number of sub-problems are much greater than the number of processors.

Selected References

Feldman J A, A distributed information processing model of visual memory, University of Rochester, TR-52.

Ballard D H, Generalizing the Hough transform to detect arbitrary shapes, University of Rochester, TR-55.

Russel D M, Constraint networks: modeling and inferring object locations by constraints, University of Rochester, TR-38.

NAVAL COMMAND AND CONTROL

Bob Bechtel
NOSC

The group I'm part of at the Naval Ocean Systems Center is concerned with the problems of naval command, control, and communication. Broadly speaking, this area includes accepting and correlating information from many diverse sources (information input), determining what is really going on (situation assessment), deciding what to do and how to do it (decision and direction), and passing on information and decisions as appropriate (report generation). We are examining possible artificial intelligence approaches to problems in these areas.

The Navy command and control structure is hierarchically organized, with nodes at one level being grouped together to form nodes at the next higher level. The lowest level that we are concerned with (in this model) is the ship. Each node, whether at ship level or higher, is faced with very similar problems. What differs among nodes is information available, resources, constraints, and possible courses of action.

To date we have explored the use of a rule-based system in situation assessment. We hope to look next at planning systems as aids in the decision and direction process. We saw distributed AI as a means of structuring, however loosely, the necessary interactions among what may eventually be several "intelligent" subsystems on a single ship, as well as providing a framework for the more physically distributed expert "communities" on several ships.

**** INFORMAL ISSUES DISCUSSIONS ****

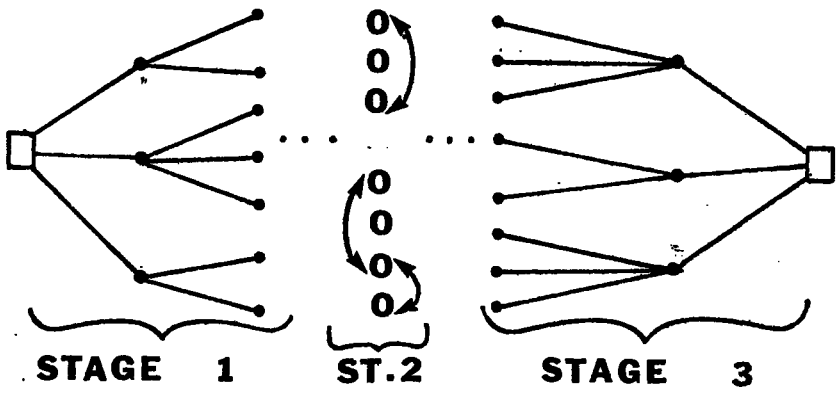
MODELS OF PROBLEM SOLVING: WHY COOPERATE?

Randy Davis

Standard problem decomposition offers a clear motivation for **distributing** a problem across a number of processors, since parallel solution of independent subproblems can increase speed. But standard decomposition does not necessarily motivate **cooperation** among the processors. In these remarks I try to isolate a few of the characteristics of a problem that make cooperation a useful and sometimes necessary approach.

I offer one way of defining what it means to solve problems cooperatively, and then examine a few traditional models of problem solving to see what kind of and degree of cooperation they employ. This helps to isolate the factors that motivate cooperation and helps make clear some of the assumptions behind standard problem solving approaches.

Consider the figure below as one way of viewing the overall problem solving process. In stage 1, the problem, initially given as a single statement, is decomposed into subproblems. In stage 2, the subproblems are solved, and in stage 3 the individual subproblem solutions are synthesized into the final answer.[1]



In these terms, one way to phrase the question is, when can the subproblems of stage 2 be solved independently, and when is communication between them required (i.e., when do all sub-task solutions have to be built with explicit knowledge of each other's progress)? This issue is traditionally referred to as the degree of coupling between the subproblems, but this only relabels the phenomenon (weakly coupled problems can be solved individually, strongly coupled can't).

---

1. In some problems the decomposition is inherent in the problem itself (e.g., the DSN), while in others there may be no need for a single answer (e.g., traffic control).

By asking a slightly different question we can perhaps get a step further in understanding what's going on and when each approach makes sense. The question is How does the failure of the subproblem independence assumption manifest itself? What goes wrong? Two obvious possibilities are

> We might not be able to solve the subproblems at all. Any partitioning of the problem yields subproblems which are simply not solvable in isolation (e.g., any subproblem alone may be ambiguous).
>
> Even if we can solve each subproblem, we might not be able to synthesize the results. That is, the solutions which result are so incompatible that no amount of patching will work.

As an example of the first phenomenon, consider the Waltz problem of understanding line drawings. If we take each vertex as a subproblem, then no subproblem is solvable by itself. We can make no progress on any solution without having the subproblems communicate.

A similar example arises in the work on speech, in both the "structural" and functional dimensions. Cutting an utterance into segments and attempting to work on each segment alone (i.e., parceling out each segment to an independent processor) permits parallelism, but typically limits very strongly the amount of progress that can be made on each segment. Allowing processors to share hypotheses as they work is a far more effective approach [1]. A similar motivation underlies the original cooperating experts model of Hearsay-II: no single expert is capable of much progress, yet the mutual sharing of information among them is a very effective technique.

To illustrate the second phenomenon (subproblems solvable, but solutions incompatible), consider a sequence of problems and traditional problem solving models. First we have those problems for which **purely modular** decompositions are possible. Examples include inference in standard logic, execution of pure lisp programs, and symbolic integration. In these cases each subproblem is solved independently and reassembly of the individual answers is trivial, so we have no difficulties.

Now consider the **almost modular** decompositions. In these cases, it is useful to proceed *as if* the subproblems are independent, and then deal with any conflicts when the individual answers are reassembled. As an example, consider some of the work on electronic circuit design [2]. A given task is divided into subtasks, each of which is solved alone. If bugs arise when the answers are assembled, the nature of the unanticipated interaction is determined and the solution "patched" to repair the problem. In this model subproblems are solved independently, and the solutions can be made compatible with some moderate amount of work. (This model is useful if there are limited kinds of interactions that can occur, and if each interaction can indeed be recognized and patched.)

Finally consider a problem like building renovation, in which many of the subproblems (deciding on the size and location of rooms for various purposes, new wiring, new plumbing, etc.) interact very strongly. Each is certainly solvable independently, but conflicts that arise when individual solutions are collected are so severe that no amount of work can make them compatible; they simply may not be patchable. In this case we can develop compatible solutions only by having the subproblems communicate and cooperate during the solution process, so that the solutions can be developed simultaneously.

The point then is simply that the independence model of problem solving can break down in (at least) two ways: the individual subproblems may be insoluble, or solutions which evolve independently may be incompatible.[2]

This provides two motivations for cooperation in problem solving, and may suggest as well the character of the information that should be exchanged. If insolubility is the problem, then each node should send its neighbor the information that most strongly reduces the neighbor's ambiguity. If compatibility is the problem, then nodes should exchange partial solutions (as in speech), underlying assumptions (building renovation, electronics design), or similar information that will ensure consistency of the eventual sub-problem solutions.

Finally, another rationale for cooperation arises out of the utility of cooperation even when subproblems *can* be solved independently. Recent work [3,4] has investigated the phenomenon in which communication between subproblems can lead to considerable speedup. Information discovered by one processor can be of sufficient use to another that the two processors can in some situations solve the problem more than twice as fast. This "acceleration effect" can be particularly useful in domains with large search spaces.

We may thus want processors to cooperate because the independence model breaks down, or because cooperation makes possible considerable speedup.

One of the themes of our work on DAI has been the focus on *what* nodes should say, not simply how they can communicate. In keeping with this, our long term intent is to continue to collect criteria like the ones above, that can help determine the content of the information to be exchanged. Eventually we hope to make it possible for cooperation between problem solvers to be structured as a series of carefully planned exchanges of information.

[1] Lesser V., Erman L, An experiment in distributed interpretation, ISI Report, April, 1979.

[2] Sussman G, Slices: at the boundary between analysis and synthesis, AI and Pattern Recognition in Computer Aided Design, Latombe, ed., 1978.

[3] Imai M, et al, A parallel searching scheme for multiprocessor systems and its applications to combinatorial problems, 6IJCAI.

[4] Kornfeld W, Combinatorially implosive algorithms, MIT AI Working paper.

---

2. Both may of course be present simultaneously. In speech understanding, for example, considering segments of the utterance in isolation often both makes them ambiguous (as noted earlier), and results in interpretations that are inconsistent when the segments are reassembled.

MODELS OF PROBLEM SOLVING
Victor Lesser

This discussion posed a number of questions and conjectures about the organization of cooperative distributed problem-solving systems:

1) Is distributed search at the heart of all distributed problem-solving structures that are robust in the face of incomplete and inconsistent information? For example, we feel the distributed Hearsay-II speech understanding system worked because it was structured as a search problem with multiple paths for deriving the correct solution. The search involves the incremental aggregation of partial solutions (with consistency relationships necessary between them) which could be derived independently or based on the context previously developed partial solutions.

2) Are the techniques developed for distributed problem-solving applicable to problem-solving in a non-distributed environment? The layering of interfaces between modules in a complex problem-solving system appear to raise exactly the same issues as having a low communication bandwidth between processing nodes in a distributed environment. It is too expensive for every node/module to have a complete and consistent view of the entire problem-solving data base. Additionally, in a system being constantly modified it may be difficult to engineer the system to be totally consistent.

3) Is the reliability (accuracy, certainty) of task processing a key parameter in deciding what types of organizational structure and control/communication policies are appropriate for a particular distributed problem-solving task? For example, is this the type of task characterization that can be used to decide whether to use an implicit, data-directed form of control (such as used in the distributed version of Hearsay-II) or a more explicit goal-directed form of control (such as used in the contract-net formalism).

4) Is there really any difference between a reductionist (decomposition) vs. a constructionist (synthesis) viewpoint in designing a distributed system?

5) Are distributed problem-solving systems by their very complexity not amenable to a single uniform control mechanism but require a number of different mechanisms for different aspects of the problem-solving?

TOWARDS A FRAMEWORK FOR DISTRIBUTED AI

Frederick Hayes-Roth

A handful of AI researchers have begun to investigate a set of domain problems, methods, and architectures for distributed and cooperative problem solving. It is not clear, from these initial forays into the new area, just what we should expect or how we could go about searching the space of issues. In this brief report, I attempt to characterize the field of Distributed Artificial Intelligence (DAI) at a high level and then suggest some promising lines of investigation. A certain shallowness is insured by the broad sweep this goal requires.

I venture a few postulates to define my prejudices and, almost surely, to provoke and elicit others to articulate alternative viewpoints. First, DAI is a subarea of AI. Nearly every major principle and practical trick of AI will apply in this new area. Second, the principal motivation for DAI is that things are, can be, or should be distributed for historical, physical or economic reasons. This distribution affects the structure, efficiency, and control of problem solving entities. This leads to the third postulate: The principles of AI should explain, in general, how to perform intellectual functions, and the principles of DAI should explain how these functions ought to be performed when aspects of them are distributed. From this point of view, the principles of DAI should articulate homologies between problem solving functions and distributed systems for representing and computing them. Most simply stated, it's the way distributed parts are organized that determines what they're good for (relatively speaking). Lastly, concerns with distribution are relativistic. Whether we choose to ignore or exploit distributions will depend on the spatial, temporal, and other dimensions of our problem, our problem solving methods, and our computing devices. Very small machines make temporal delays an important component of problem solving; limited bandwidth for communication makes informative exchange desirable; and so forth. ALL REAL SYSTEMS ARE DISTRIBUTED.

Admittedly the preceding was pretty abstract, but DAI has many concrete motivations and entailments. If we think of AI principles as relations among problem types, solution methods, and implementation architectures, the interest in DAI arises because each of these three components has natural distributions. Hence, we search for ways to distribute the other two components so that a whole problem solving system exploits the potential benefits of distribution. Most familiar AI problem domains (e.g., planning, vision, diagnosis) have characterizations that distribute states or goals over sets that permit parallel search and evaluation (e.g., actions and events in the future are distributed over space and time; pixel interpretations are distributed over label sets and <x,y> coordinates; and diagnosis problems collect data distributed over time and space and explain them by mapping them onto appropriately distributed models).

In addition, both problem solving methods and architectures have natural distributions. For example, the Hearsay-II architecture was designed, in part, to exploit asynchronous distributed computers in support of a particular hypothesize-and-test activity. In fact, the speech system never exploited this potential capability. On the other hand, at the level of problem solving method, the Hearsay-II speech system distributed its knowledge among independent knowledge source specialists that distributed their hypotheses over a globally accessible memory with time and linguistic abstraction dimensions. These distributions of the problem solving function reflected homologous distributions of the speech problem itself, and several experiments suggest that profitable mappings between the problem solving function and underlying distributed architectures can also be developed.

If distribution of parts (in problems, methods, and architectures) underlies DAI, surely we must develop a much improved understanding of what the parts are, how they can be distributed, and what it takes to integrate and coordinate them. In my initial survey of distributions, I have noted that distributions act as functions from systems to systems and their effect is to replace one or more system components with sets of components that are ordered over a domain. The six primary domains for DAI distributions seem to me: (1) space, (2) time, (3) instrumentality, (4) resources, (5) information, and (6) interpretation. These domains provide more or less natural bases for organizing, partitioning, and ordering individual components into distributed sets. The trick in DAI is to develop and choose bases that enable us to distribute computing without inordinately increasing computational or physical complexity.

In sum, distributing the components of a system introduces new levels of organization. Distributions derive value from their ability to exploit opportunities for parallelism. Their value diminishes when they introduce intrinsic complexity or if they impede other problem solving methods or architectures that are already well adapted to the problem requirements. Finally, predicting the effects of arbitrary distributions will not be possible without a much deeper understanding of the relationships between problem, solution, and computational components. Because these too may change relatively quickly (new machines possible in months, new collections of cooperating experts in hours), we should anticipate an extraordinary range of interesting, perhaps surprising, phenomena. I suspect the greatest payoffs in the short-run in applications to distributed planning and surveillance, while in the longer run I anticipate knowledge storage and retrieval will be the most important application of DAI.