An Overview of DAI: Viewing Distributed AI as Distributed Search[1]

Victor R. Lesser
Computer and Information Science
University of Massachusetts/Amherst

Introduction

One approach to understanding Distributed AI is to view it as a distributed state space search. A distributed search involves partitioning the state space and its associated operators and control regime so that multiple processing elements can simultaneously perform local searches on different parts of the state space; the (intermediate) results of the local searches are shared in some form so that the desired answer is produced in a timely manner. The requirements for an effective distributed search may necessitate a reorganization of the state space search in ways which would not be optimal for a sequential search. Research in DAI emphasizes the development of approaches for partitioning the search space into a set of local searches and coordinating these simultaneous searches in terms of both information sharing and control. In turn, the granularity and form of the partitioning of the state space search, the hardware characteristics of the processor organization which implements this distributed search (independence and complexity of processing elements, the connectivity patterns among elements, and relationships among communication and processor bandwidths), and the distribution of expertise and information in the processor organization engender very different sets of issues for DAI researchers.

On one extreme are researchers who study fine-grain decompositions. Here the state space search is partitioned into local searches which involve small collections of states and where simple computations are sufficient to implement the operators and control. An example of this type of partitioning is Waltz's constraint satisfaction network or a relaxation algorithm [30, 31]. In this case, the classic state based search is reorganized. Instead of repre-

[1]This work was supported by the Office of Naval Research, under grant number N00014-89-J-1877, and University Research Initiative grant number N00014-86-K-0764.

senting a state in the search space as a list of distinct attribute value pairs which define a complete (or partial) path in the search for an acceptable goal state, there is a distinct state, called a node, for each attribute. Each node contains the current set of allowable values for the attribute; a local search process is associated with each node. This fine-grain distributed search could be matched with a fine-grain processor organization where each local search process is associated with one processing element; processors are simple and there can be a very large number of them; they work in a lockstep manner with other processors and can communicate in fixed patterns with "nearby" processing elements at processor speeds. Researchers studying this form of DAI call themselves connectionists and have separated themselves from other DAI researchers since their concerns are so different.

At the other extreme are researchers who study large-grain decompositions. In this case, the state space search is partitioned into a relatively small set of dependent and independent subproblems (tens to hundreds) in a form somewhat analogous to how a state space search can be alternatively represented in terms of an AND/OR problem reduction graph. In this case, the problem reduction graph is partially expanded to a level where each terminal subproblem requires significant problem solving. This large-grain distribution of problem-solving activity could be matched with a large-grain processor organization where processors are complex (e.g., a Lisp processor) and operate asynchronously. Groups of subproblems are partitioned among processing elements. These processors could use very sophisticated strategies to coordinate their problem-solving activities and communications with other processing elements.

Alternatively, there are hybrid situations where the granularity of the decomposition of the search space does not match the granularity of the processor organization. For example, the Actor framework of Hewitt [17] involves small-granularity activity; these agents are simple asynchronous processes with rudimentary message-passing capabilities. The implementation, however, has been on a large-grain and loosely-coupled processor organization where large numbers of agents are assigned to each complex element.

The majority of research in DAI involves large-grain search space decompositions on large-grain processor organizations. This particular model of distribution in DAI has come to be called Distributed Problem Solving (or Cooperative Distributed Problem Solving) and will be the focus of this overview article.

Cooperative Distributed Problem Solving

Cooperative Distributed Problem-Solving (CDPS) networks are broadly defined as loosely-coupled distributed networks of semi-autonomous problem-solving nodes that perform sophisticated problem solving and cooperatively interact with other nodes to solve a *single* problem [12]. Nodes cooperatively solve a problem by individually solving subproblems (that are possibly interdependent and overlapping) and integrating the subproblem solutions into an overall solution. Solution integration does not require that the solution is completely represented at any one node; in some situations, components of the solution could be distributed across the network, integrated only by their mutual consistency. Due to the distribution of expertise and information in the network, often a node cannot completely solve a subproblem without assistance from other nodes. In the face of limited communication bandwidth, this leads to a local problem-solving strategy where nodes "do the best they can with available knowledge." Nodes cooperate by generating and exchanging tentative and partial results with each other until sufficient information has been exchanged to construct mutually consistent, complete solutions to the local subproblems.

In a CDPS network, each node can modify its behavior as circumstances change and can plan its own communication and cooperation strategies with other nodes. Nodes operate asynchronously and in parallel with other nodes with *limited internode communication.* Internode communication is either constrained by inherent bandwidth limitations of the communication medium or by the high computational costs of packaging and assimilating information to be sent and received among nodes.

CDPS differs significantly from distributed processing. A distributed-processing network typically has multiple, disparate tasks executing concurrently in the network. In this case, shared access to physical or informational resources is the main reason for interaction among tasks. The goal is to preserve the illusion that each task is executing alone on a dedicated system by having the network-operating system hide the resource-sharing interactions and conflicts among tasks in the network. In contrast, the problem-solving procedures in CDPS networks work together to solve a single problem. These procedures are explicitly aware of the distribution of the network components and can make informed interaction decisions based on that information. Unlike CDPS networks, where cooperation among nodes is crucial to developing

a solution, the nodes in traditional distributed-processing applications rarely need the assistance of another node in carrying out their problem-solving functions.

CDPS has become an important research focus in DAI for several reasons. First, advances in hardware technology for processor construction and for processor communication make it possible to connect large numbers of sophisticated, yet inexpensive, processing units that execute asynchronously. The use of interconnected processors can be a cost-effective means of providing the computational cycles required by AI applications. A range of connection structures are possible, from a very tight coupling of processors through shared or distributed memory, to a more loose coupling of processors through a local area communication network, to a very loose coupling of geographically distributed processors through a long-haul communication network.

Second, there are many AI applications that are inherently distributed. The applications may be spatially distributed, such as interpreting and integrating data from spatially distributed sensors or controlling a set of robots that work together on a factory floor. The applications may be functionally distributed, as when bringing together a number of specialized medical-diagnosis systems on a particularly difficult case, or in developing a sophisticated architectural expert system composed of individual experts for structural engineering, electrical wiring, room layout, etc. The applications may be temporally distributed (pipelined), as in a factory application where there is a production line consisting of a number of work areas each with an expert system responsible for scheduling orders. A problem-solving architecture that matches the distribution of data, expertise, and processing power has significant advantages over a single, monolithic, centralized process in terms of processing and communication efficiency, reliability, and real-time responsiveness.

Third, the ability to structure a complex problem into relatively self-contained processing modules leads to systems that are easier to build, debug, and maintain than a single monolithic module and that are more resilient to software and hardware errors.

Finally, understanding the process of CDPS is an important goal in its own right. It is possible that the development of CDPS networks may serve the same validating role to theories of sociology, management, and organizational theory, as the development of AI systems have served to theories of

problem solving and intelligence in linguistics, psychology, and philosophy.

Issues in Cooperative Distributed Problem Solving

The major intellectual issues in CDPS, and more generally in DAI, arise when it is not possible to decompose problem solving into a set of subproblems so that there is a perfect fit between where information, expertise, processing, and communication capabilities lie in the processor network and the computational needs for effectively solving each subproblem. An additional issue that can arise in the structuring of CDPS architectures is the requirement for high reliability.

In the best of all possible worlds, problem solving would be decomposed into a fixed set of independent subproblems with each requiring approximately the same amount of computation; the associated processor organization would have exactly the same numbers of processors as subproblems so that each subproblem could be assigned to a unique processor; each processor would also have the requisite processing capabilities, information and expertise so that each subproblem could be solved completely based only on local problem solving. This type of optimal decomposition would keep all the processors busy doing effective problem solving. Unfortunately, it is very difficult to find real-world applications that exhibit such an optimal mapping between the problem-solving decomposition and the processor organization. In most applications, the mapping leads to situations where one or more of the following is true: 1) there are more subproblems than processors, subproblems require different amounts of computation to solve and new subproblems may be created during problem solving; 2) subproblems on different nodes are interdependent, requiring nodes to coordinate their local problem solving so that consistent solutions will be constructed; 3) information and expertise is distributed in the network so that some processor nodes have insufficient information or expertise to completely solve a subproblem locally and the cost of transferring all the necessary expertise and information to the appropriate node is prohibitively expensive in terms of time delay due to limited communication bandwidth; and 4) significant amounts of computation and communication are required to synthesize an-

swers from the individual subproblem solutions and distribute these answers to the appropriate nodes. To illustrate some of these issues, specifically 1 and 2, consider a simple distributed search.

The example problem is a planning task where the initial state is W_O and the goal state is defined in terms of the following predicates: $A(x) \wedge B(y) \wedge C(u) \wedge D(v) \wedge E(w)$; there is also the constraint that the $x + y = z$, where z is a constant. Suppose each node in the network is initialized with a copy of W_O and there is sufficient expertise at each node to solve any of the subgoals. If we are to solve subgoals $A(x)$ and $B(y)$ on separate nodes in the network, then we have to make sure that the separately derived solution of x_i for $A(x)$ on node$_k$ and solution y_j for $B(y)$ on node$_l$ are compatible; solutions can be incompatible because either the simultaneous achievement of $A(x_i)$ and $B(y_j)$ leads to world states that cannot be merged into a consistent world state or because $x_i + y_j \neq z$. The recognition of incompatible solutions to subgoals is difficult because of limited communication bandwidth, which may make it impossible to transmit all the changes to the world state required in achieving subgoals on one node to all other nodes with interacting subgoals. A cooperative strategy employing limited communication would be for nodes to exchange an abstracted view of the changes to their world states so that an incompatible world state could be recognized without significant communication or some type of exchange of high-level partial results by nodes during problem solving could be developed so that incompatibilities could be recognized early and without much communication [4].

The problem of guaranteeing that the constraint $x+y = z$ is met, makes it difficult to solve these subgoals independently for similar reasons, though the recognition of constraint violations may require less communication because they often do not involve a lot of state information. As with sequential search, these constraints can also be used to reduce search by constraint propagation. For example, if a result of partial problem solving of node$_l$ working on $A(x)$ can bound x such that $x_l \leq x \leq x_u$, then this bound, if transmitted to node$_k$ working on $B(y)$ will speed up processing since $(z - x_l) \leq y \leq (z - x_u)$.

Another major issue involves effectively searching the goal tree in parallel; the development of a distributed schedule for the ordering of the achievement of subgoals on individual nodes is also complicated by limited communication. In order to perform the distributed search efficiently, the distributed schedule must take the following factors into account: the potential for parallelism, the current tasks being performed by nodes, and interdependencies

among solution subgoals. For example, suppose node_1 has subgoals A, C and E to achieve while node_2 has subgoals B, D and E to achieve, where only A and B are interacting through constraints. If, for instance, the solution to subgoal A was both more constraining and faster to compute, then an appropriate scheduling strategy would be for node_1 to first do subgoal A and node_2 to first do subgoal D, such that when node_2 starts to work on subgoal B, it will be able to solve it within the constraints defined by the solution to subgoal A. This strategy has two advantages: it avoids unnecessary work in achieving subgoals in ways that will eventually be incompatible, and it makes the search for a solution to the constrained subgoals (in this case, subgoal B) more efficient. Thus, if the distributed scheduler can recognize potential interactions among subgoals, and there is some freedom about the order in which subgoals can be achieved, a more efficient search can be conducted. Another issue in distributed scheduling, exemplified by the previous example, is the avoidance of redundant computation (e.g., both nodes have E as a subgoal to achieve). Again, avoiding redundant computation poses a problem, but also opportunities. The opportunities involve: more effective load balancing since there is more than one node that can perform a computation; higher reliability, since if one node fails there is still a possibility that the solution to its subgoals could be derived at another node; and, increased parallelism if nodes having the same subgoals pursue different approaches to solving the subgoal. For example, if subgoal E, that both nodes 1 and 2 can solve, could be further decomposed so it can be achieved by either solving subgoal $E1$ or subgoal $E2$, then disjunctive parallelism speedup could be achieved if node 1 attempts to solve E by solving $E1$ while node 2 solves E by solving $E2$.

The requirement for determining whether a consistent and optimal set of solutions to the top-level subgoals has been constructed also presents problems in a distributed setting. Thus far we have not considered optimality in our discussions, but often there are additional parameters associated with goals such as confidence in the answer or the amount of resources used. The concerns for optimality require the termination criterion to closely look at what alternative solutions to goals have already been explored, what options can be further explored, and what the potential quality is of these alternatives. Again, the difficulty is making these decisions distributively, without access to all the details of the current state of the network search.

The problem of effectively dealing with subgoal interactions becomes even

more complex when operators on different nodes generate conflicting solutions to the same or interacting subgoals because of differences in their underlying world view. These differences can arise when nodes have the same long-term knowledge and problem-solving strategies, but have access to different input data. For example, nodes that get their input data as a result of sensors could have different data due to sensor error or due to sensors being more or less sensitive to phenomena in the environment because of their geographical placement (one sensor cannot observe a vehicle moving because it is blocked by a mountain) or because of different sensor types (acoustic versus radar). These differences also arise because nodes may use different sets of operators which have inconsistent long-term knowledge or different problem-solving strategies that lead to alternative parts of the search space being examined. For example, in a multi-criteria problem such as building design where criteria for a successful design involves both structural safety and cost considerations, one node could have long-term knowledge that emphasizes criteria that generate solutions where the building is very safe, while another node's long-term knowledge may emphasize criteria that lead to buildings that are very cheap to construct. Another example is where one node's control strategy emphasizes getting an acceptable solution quickly while the other node's emphasizes getting the optimal answer. One final example, which combines issues of inconsistency in both long-term and short-term knowledge would be in cooperative medical diagnosis, where each expert will try to diagnose the illness in terms of its area of specialized medical knowledge (long-term knowledge) and in terms of cases it has recently seen (short-term knowledge).

From this "simple" example, we can get some insight into the issues that arise when the mapping between the problem decomposition and processor organization is not optimal. Thus, the importance placed on CDPS research on the development of the following capabilities:

- Distributed network control protocols — This involves dynamically assigning newly generated subproblems to appropriate agents, reorganizing the assignments of subproblems to agents in the face of changing workloads and processor/communication configurations and scheduling of local subproblems on each agent so that concerns for efficient network problem solving are integrated with those of efficient local problem solving;

- Protocols for cooperative problem solving among agents through sharing of information and expertise — This requires techniques for exploiting interdependencies among agents' subproblems to assist local problem solving so as to make up for incomplete and inconsistent information and expertise, and to speed up processing; it also involves techniques for resolving inconsistencies among both agents' short-term and long-term knowledge and beliefs; and

- Local problem-solving architectures that can effectively operate in a CDPS context — Agents must be able to accomplish reasonable problem solving in the face of possibly incomplete, inconsistent and out-of-date information and expertise and to make informed communication decisions by requesting information that would be helpful and communicating information that is important to other agents. Additionally, agents may be required to have a high-level model of their current and past problem solving and to make predictions about their future activity so as to provide information necessary for effective network control. Finally, effective network control may require agents to reorganize how they schedule and perform their local problem-solving tasks.

These techniques cannot depend on always having a highly detailed, consistent and up-to-date view of the state of local problem solving throughout the network [23] due to the lack of adequate communication bandwidth, the time delays introduced in end-to-end communication, the desire for these techniques to be robust in the face of hardware failures and the potential for significant decreases in exploitable parallelism in the network if the activities of agents are tightly synchronized. Often, the techniques that have been developed trade off lowered network communication for increased amounts of local computation. More computation occurs because the information required to make informed control decisions is not always available or up-to-date nor does the information being used always lead to correct or complete inferences; this creates local problem solving inefficiency since more search is required to reach an acceptable answer or partial result. This extra search can be both local and network wide. If there are sufficient constraints on the local acceptability of answers, inferences based on faulty or incomplete information will never lead to the generation of an incorrect answer. However, in the case where there is insufficient local criteria to eliminate incorrect answers, these incorrect results need to be eliminated as a result of network

problem solving; thus, local problem solving may be revised as a result of information received from other agents. The other method used to trade off less communication for more local computation is to use more sophisticated local problem-solving techniques to enable agents to transmit a more informed and higher-level view of the consequences of their local problem, and also to give agents the ability to exploit their own information and that received from other agents more fully in their local problem solving.

In general, the approaches discussed above shift the emphasis from optimizing the efficiency of individual problem-solving activities to achieving an acceptable performance level (in terms of quality of the answer, the robustness of the system, and requirements for communication and processor resources) for the network as a whole. Implicit in this satisficing approach [25] to network problem solving is the recognition that it is very difficult to create an optimal mapping throughout the lifetime of problem solving between the problem-solving decomposition and the processor organization.

State of the Art

Research in DAI is still very much in its infancy [7]. Over the last ten years, a certain level of understanding of important conceptual issues in structuring DAI systems has been developed, and a collection of interesting but rather disparate sets of approaches, frameworks, and specific techniques has been partially evaluated. To date, a deep conceptual or theoretical framework for comparing alternative approaches has not been developed. These alternative approaches often make different and highly specific assumptions about the underlying problem-solving structure and processor organization, so it is difficult to compare or reproduce results on slightly different problems. From a more positive viewpoint, DAI research has developed the following important concepts for the design of DAI systems.

1. Organizational structuring: The use of predefined roles and communication patterns for agents as a way of organizing problem solving. Organizational structuring reduces the complexity of agents' control problem solving by constraining the range of dynamic control decisions. A balance can be achieved so that coordination requires less overhead but still leads to acceptable, though not necessarily optimal, network problem-solving performance. [5, 13, 14, 18, 20]

2. Negotiation: The use of a cooperative dialogue among agents to resolve the uncertainty and incompleteness of an agent's knowledge base. These are highly-directed dialogues that attempt to minimize the communication necessary to arrive at consistent solutions to interdependent subproblems and implement network control. [1, 3, 6, 10, 21, 22, 27, 28, 29]

3. Tolerance of inconsistency: An agent's problem solving is structured to operate with local knowledge bases that can be incomplete, inconsistent, and out-of-date. Error resolution is an integral part of network problem solving and agents do the best they can with their current information. [9, 18, 23, 24]

4. Sophisticated local control: The more sophisticated an agent is about its problem-solving plans, the status of its beliefs, and the implications of its actions on other agents' plans and beliefs, the easier it is to achieve coordinated behavior among agents. A corollary of this viewpoint is the importance of the interplay between local and network control in order to achieve effective network problem solving. [5, 8, 11]

5. Reasoning about other agents: To make informed problem-solving and control decisions, an agent needs to be able to acquire, represent, and use beliefs and intentions of other agents. [2, 15, 16, 19, 26]

These five concepts are interrelated and aspects of each appear in most DAI architectures. They lay the groundwork for future systems and research in DAI.

DAI seems very appropriate to the next generation of applications and computing organizations that are now in the planning stages. These applications will be structured as complex, distributed problem-solving systems in which both man and machine are participants. This next generation of systems will also provide challenges to DAI researchers since they will be heterogeneous, be required to operate under both soft and hard deadlines, need to be robust, and may be very large. The issue raised by these requirements have barely been touched in DAI research. Thus, there are many exciting research and implementation challenges facing DAI researchers, such as creating real-world systems that exploit existing ideas, developing conceptual frameworks that integrate the experiences gained already, and expanding

our research focus to include the problems faced by this next generation of applications.

Acknowledgments

I want to acknowledge the importance of my collaborative research with colleagues in shaping the views expressed in this paper. Each of my colleagues has contributed important insights, beginning with my first published article on DAI with Lee Erman, through the long-term joint research efforts with Daniel Corkill and Edmund Durfee, and continuing in my recent work with Susan Conry, Robert Meyer, Kazuhiro Kuwabara, Keith Decker, Susan Lander, Brigitte Maître, and Hassan Lâasri.

References

[1] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 767–770, Karlsruhe, Federal Republic of Germany, August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 102–105, Morgan Kaufmann, 1988.).

[2] Philip R. Cohen and Hector J. Levesque. Intention = choice + commitment. In *Proceedings of the National Conference on Artificial Intelligence*, pages 410–415, Seattle, Washington, July 1987.

[3] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufmann, 1988.

[4] Daniel D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 168–175, Cambridge, Massachusetts, August 1979.

[5] Daniel D. Corkill and Victor R. Lesser. The use of meta-level control for coordination in a distributed problem solving network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, Karlsruhe, Federal Republic of Germany, August 1983. (Also published in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986).

[6] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, pages 63–109, 1983.

[7] Keith S. Decker, Edmund H. Durfee, and Victor R. Lesser. Evaluating research in cooperative distributed problem solving. In *Distributed Artificial Intelligence*, volume 2, pages 487–519. Pitman Publishing, Ltd., London, England, 1989.

[8] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 875–883, Milan, Italy, August 1987. (Also published in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser, editors, pages 281-293, Morgan Kaufmann, 1988.).

[9] Edmund H. Durfee and Victor R. Lesser. Predictability versus responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the National Conference on Artificial Intelligence*, pages 66–71, St. Paul, Minnesota, August 1988.

[10] Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence*, volume 2, pages 229–244. Pitman Publishing, Ltd., London, England, 1989.

[11] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C(11):1275–1291, November 1987. (Also published in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser, editors, pages 268-284, Morgan Kaufmann, 1988).

[12] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–82, March 1989. (Invited paper).

[13] Mark S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):70–80, January 1981. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 140–150, Morgan Kaufmann, 1988.).

[14] Les Gasser, Nicolas Rouquette, Randall W. Hill, and John Lieb. Representing and using organizational knowledge in distributed ai systems. In *Distributed Artificial Intelligence*, volume 2. Pitman, London, England, 1989.

[15] Barbara J. Grosz and Candace L. Sidner. Discourse structure and the proper treatment of interruptions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 832–839, Los Angeles, California, August 1985.

[16] J. Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. IBM Research Report IBM RJ 4421, IBM, 1984.

[17] Carl Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, Fall 1977.

[18] Carl Hewitt. Offices are open systems. *Communications of the ACM*, 4(3):271–287, July 1986. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 321–330, Morgan Kaufmann, 1988.).

[19] Kurt Konolige. A deductive model of belief. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 377–381, Karlsruhe, Federal Republic of Germany, August 1983.

[20] William A. Kornfeld and Carl E. Hewitt. The scientific community metaphor. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):24–33, January 1981. (Also published in *Readings in Distributed*

Artificial Intelligence, Alan H. Bond and Les Gasser, editors, pages 311–320, Morgan Kaufmann, 1988.).

[21] Kazuhiro Kuwabara and Victor R. Lesser. Extended protocol for multistage negotiation. In *Proceedings of the 9th Distributed Artificial Intelligence Workshop*, pages 129–161, 1989.

[22] Susan Lander and Victor Lesser. Negotiation among cooperating experts. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[23] Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–96, January 1981.

[24] Victor R. Lesser and Lee D. Erman. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, C-29(12):1144–1163, December 1980. (Also published in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser, editors, pages 120-139, Morgan Kaufmann, 1988.).

[25] James G. March and Herbert A. Simon. *Organizations*. John Wiley & Sons, 1958.

[26] Stan Rosenschein. Reasoning about distributed action. *AI Magazine*, 84:7, 1983.

[27] Arvind Sathi and Mark S. Fox. Constraint-directed negotiation of resource reallocations. In *Distributed Artificial Intelligence*, volume 2, pages 163–193. Morgan Kaufmann, California, 1989.

[28] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980.

[29] Katia Sycara. Resolving goal conflicts via negotiation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 245–250, St. Paul, Minnesota, August 1988.

[30] David Waltz. Understanding line drawings of scenes with shadows. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.

[31] S. W. Zucker, R. A. Hummel, and A. Rosenfeld. An application of relaxation labeling to line and curve enhancement. *IEEE Transactions on Computers*, C-26:394–403, 922–929, 1977.