Evaluating Research in Cooperative Distributed Problem Solving

Keith S. Decker Edmund H. Durfee
Victor R. Lesser [1]

UMass Computer Science Technical Report 88–89
August 1988

Abstract

[1] This research was sponsored, in part, by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

Cooperative Distributed Problem Solving (CDPS) is a new field that has not yet established critical research mass, so relating and assessing individual research contributions is difficult. In this paper, we provide a framework for understanding the interrelationships of disparate research efforts in order to assess progress in the field. We evaluate progress in these basic issues by assessing the sophistication of CDPS systems on many different (and interdependent) axes, which when taken as a whole give a view of a system's overall sophistication. Our sophistication assessments are specified as a set of questions that relate to such things as the class of problem domains and environments that an approach can handle, the theoretical soundness of the approach, the software and hardware support provided, and how the approach addresses the important issues of CDPS themselves.

We explain the general goals of CDPS research, present questions pertinent to them, and describe a few examples of CDPS research: the contract net, partial global planning, and proposed extensions to partial global planning. We evaluate these systems and discuss how the questions relate to the multiple goals of CDPS research, and how this methodology can be used to analyze the differences between the above systems.

1 Introduction

Cooperative Distributed Problem Solving (CDPS) is a new field that has not yet established critical research mass, so relating and assessing individual research contributions is difficult [5, 2]. In this paper, we provide a framework for understanding the interrelationships of disparate research efforts in order to assess progress in the field.

We broadly define cooperative distributed problem solving networks as loosely-coupled distributed networks of semi-autonomous problem solving agents (or nodes). These agents each perform sophisticated problem solving and cooperatively interact to solve problems. When faced with a problem that could be solved more effectively through cooperation (perhaps because it requires expertise or information localized in different agents), the agents work together by identifying subproblems each should solve, solving them concurrently, and integrating their results (or at least ensuring that their results are mutually consistent). In most complex problem solving applications, subproblems are interdependent and overlapping, so agents must carefully coordinate their local problem solving actions, and must modify how they interact as circumstances change in the course of problem solving. This dynamic coordination must occur despite internode communication that is limited due to inherent bandwidth limitations or to the high computation costs of packaging and assimilating transferred information.

The basic issues that must be addressed in CDPS research include developing:

- theories for organizing CDPS systems in both small groups and large networks, providing guidelines for how domain and control problem solving tasks need to be distributed among agents based on the characteristics of the application, the agents, and the communications resources;
- paradigms for domain-level cooperation among disparate agents that can efficiently resolve inconsistencies and integrate results;
- methods for distributed control, permitting semi-autonomous agents to work in a globally coherent manner despite uncertainty about the status of other agent's activities;
- guidelines for organizing or structuring the problem solving activities of individual agents so that they may be easily integrated into a CDPS environment,

leading to novel problem solving frameworks (knowledge representations, inference techniques, and control architectures) that are more appropriate for use in CDPS systems than traditional techniques are;

- software infrastructures, in terms of languages and operating system features, that support the CDPS style of computing.

These issues center around how to achieve effective cooperation that balances the interdependent criteria of efficiently using processor and communications resources, reliability, responsiveness to unexpected situations and real-time deadlines, and modularity. Effective cooperation requires resolving the uncertainties and inconsistencies that can arise between both long-term and short-term knowledge held by agents and exploiting the interdependencies among the agents' subproblems.

We evaluate progress in these basic issues by assessing the sophistication of CDPS systems along many different (and interdependent) axes, which when taken as a whole give a view of a system's overall sophistication; we view increasing sophistication as an indication of progress. Our sophistication assessments are specified as a set of questions, where each question has a set of answers that range from less to more sophisticated. Because the questions are subjective, and any strictly numerical means of recording, normalizing and summarizing the answers would result in a false sense of precision, we divide each set of answers into only three levels of sophistication (low, medium, and high). These questions relate to such things as the class of problem domains and environments that an approach can handle, the theoretical soundness of the approach, the software and hardware support provided, and how the approach addresses the important issues themselves. The questions are grouped into general goals, but each question often applies to more than one goal or issue. Since most research deals only with a subset of all the possible issues, not all questions may be applicable to a given theory, environment, or domain.

Our evaluation technique cannot be used to compare two entirely different research programs, since each will have different goals. Furthermore, if the goal of a project is to solve a specific domain problem, then extra sophistication is not always useful or welcome. This technique is most profitably used as a guide to whether a research program addresses *any* important aspects of CDPS, and as a guide to the progress of a particular research program over time.

Sections 2–10 explain the general goals of CDPS research and contain questions pertinent to them. Section 11 describes a few examples of CDPS research: the contract net [10, 1, 11], partial global planning [3], and proposed extensions to partial global planning. Section 12 evaluates these systems and discusses how the questions relate to multiple goals, and how this can be used to analyze the differences between the above systems.

2 Goal: Limit Domain and Environmental Assumptions

One goal of CDPS research is to limit the number of assumptions that a CDPS theory has to make about the environment, or a CDPS architecture has to make about the problem domain. Even if the system was developed to solve a specific problem, the less assumptions made, the more generally applicable the system is to CDPS problems. The following questions pertain to the heterogeneity of the system, structural assumptions (about how the system is put together), and other domain and architectural assumptions. So the less assumptions that are made about the domain or the way the system is put together, the higher the rating.

2.1 How heterogeneous are the agents in your system?

Low: They are identical, or differ only in the resources available to them.

Med: They differ in the problem solving methods or expertise available at each node.

High: They share a common language, but no other assumptions have been made about the nodes.

2.2 What assumptions are made about the maximum number of agents in your system?

Low: There must be less than ten[1].

Med: There must be less than a hundred.

High: There can be more than a hundred.

[1]These numbers are derived from our experience with the DVMT.

2.3 How is problem solving knowledge shared among agents?

Low: Agents have identical knowledge bases.

Med: Agents' knowledge may differ, but mechanism/structure of knowledge is same (all frames or all logical assertions, etc.). Agent knowledge can be disjoint or overlapping.

High: Agents' knowledge is not just different, but may be *inconsistent* between agents. The structure of knowledge differs between agents. Algorithms or procedures may be entirely different.

2.4 How is short term knowledge and data that is acquired during problem solving replicated among agents?

Low: Each agent has identical short-term knowledge (data).

Med: Short-term knowledge (data) differs, and may be disjoint or overlapping.

High: Short-term knowledge (data) may be inconsistent between agents, or the representation/structure of knowledge is different at each agent.

2.5 How is the consistency of short term problem solving knowledge and data maintained?

Low: Cannot handle inconsistent short term knowledge or data, or provide concurrent access and maintain consistency by distributed database techniques.

Med: When necessary, can invoke a conflict resolution mechanism — inconsistencies are an extraordinary process in the course of problem solving.

High: Allow inconsistencies to occur and resolve them as a part of normal problem solving without a separate conflict resolution process.

2.6 How is the consistency of long term problem solving knowledge and data maintained?

Low: Cannot handle inconsistent long term knowledge or data, or provide concurrent access and maintain consistency by distributed database techniques.

Med: When necessary, can invoke a conflict resolution mechanism — inconsistencies are an extraordinary process in the course of problem solving.

High: Allow inconsistencies to occur and resolve them as a part of normal problem solving without a separate conflict resolution process.

2.7 Is the first solution obtained the correct one?

Low: Yes, it is correct and optimal.

Med: Either the system must obtain all solutions, or the system includes a mechanism for deciding when additional processing will not lead to a superior solution.

High: System attempts to trade-off optimality of a solution and other considerations, such as meeting a deadline.

2.8 Do the agents make any assumptions about the operating system?

Low: Have developed, and are dependent on, OS and hardware support, or have developed a new OS for the agent environment.

Med: Uses the OS supplied by the manufacturer in a non-portable way..

High: No assumptions made — OS independent (high on the domain independence scale but low on sophisticated system support scale).

2.9 Can the agents interact with humans as agents?

Low: No human interaction during problem solving

Med: Some agents interact with humans in standard computer interface ways, but the humans are there only to give input to an agent and receive output.

High: The system is a *participant system*. Humans are actual agents, in that they can interact with computer agents just as other computer agents do. They appear as computer agents to each other and to the real computer agents.

2.10 Do you assume that new agents or communication paths can be dynamically added and deleted during problem solving?

Low: No, not dynamically.

Med: Communications paths, but not new agents.

High: New agents and new communication paths can be added and deleted dynamically.

2.11 Is problem solving continuous or discrete?

Low: System acts on discrete problems.

Med: System is in continuous operation, but individual problems are not related and do not usually occur simultaneously.

High: System is involved in continuous domain and control problem solving.

2.12 Has the system been evaluated in multiple domains?

Low: One domain.

Med: Two domains.

High: More than two domains.

3 Goal: Discover Paradigms for Building Cooperating Agents

We wish to build systems where agents can *cooperate* with one another to solve problems. This does not imply that the agents necessarily share the same set of goals, but rather that agents can coordinate effectively to approach common goals when they or their designers deem such actions appropriate. The questions here deal with interacting goals and conflicts between agents. The more the domains being pursued require cooperation, and the more the system supports it, the higher the rating (the more "sophisticated").

3.1 How do the goals of your agents interact?

Low: Agents do not have goals, and interactions between agents are pre-specified. Alternatively, each agent has separate, non-interacting goals.

Med: The system of agents is constructed with a shared set of high-level goals (the benevolent agent assumption was made). An agent's goals are derived from these. The low-level goals of two agents may constrain one another critically, but each agent can locally deal with such interactions.

High: Agents goals may interact destructively, and agents must interact with one another (or within some organizational structure) to resolve conflicts. A high-level goal may be shared serendipitously — agents recognize when goals interact and work towards positive outcomes of such goals (when this is possible and desirable) or try to outmaneuver and foil another agents goals (when this is possible and desirable).

3.2 Do you represent and reason about an agent's goals?

Low: Agents do not have goals.

Med: Agents' goals are implicit.

High: Agents' goals are explicitly represented and there are processes that make decisions based on goal interactions.

3.3 What range of collaboration is possible between agents?

Low: Agents act independently of one another, or in a fixed relationship.

Med: Solutions to one agent's subproblem may help, predict, or critically constrain the solution to another agent's subproblem.

High: Not only are there critical inter-agent constraints, but critical timing constraints in the actions of the agents. There may be undoable or non-reversible actions.

3.4 How are domain conflicts resolved?

Low: There are no conflicts or conflict resolution.

Med: Agents use a mutually known decision procedure or an arbitrator.

High: Agents negotiate by loosening of constraints of one or both agents. Agents may choose not to resolve some conflicts, or to use other methods if appropriate to the situation.

3.5 How do agents deal with non-reversible actions?

Low: Agent actions are always reversible, or any action that was possible in the past is always possible in the future (so there is no need to reverse actions).

Med: The agent handles the situation locally or appeals to a higher authority (other agent).

High: The agent will, depending on the situation, appeal to other agents or carry out the action and recover from the consequences. The agent tries to predict and avoid problems with non-reversible actions.

3.6 Can the prototypical problems (for which the system was designed) be solved without cooperation?

Low: Yes; although prototypical problems involve multiple agents, a centralized agent is used to solve them.

Med: Yes, but there are significant organizational, efficiency or fault tolerance reasons for solving them in a distributed fashion.

High: No, for some theoretical reason (bounded rationality, for instance) cooperation is necessary.

4 Goal: Develop Methods for Assuring Global Coherence

We wish to develop systems of agents that can react in a reasoned manner to the varied constraints of local, global, and partially global goals and input data. We wish to look at problems where an agent's local point-of-view crucially affects how that agent sees the global problem situation, and systems that can solve such problems. Global coherence addresses the problem of how members of a group can make sensible decisions that lead to problem solutions with only a limited amount of information. The questions in this section cover the types of information used and their availability. A system receives a higher rating if achieving a total, global view of the problem is not attempted or is impossible, and if an entirely local view is also insufficient for satisfactory results.

4.1 What is the range of non-local information that agents use to assure coherent behavior?

Low: No nonlocal information is used.

Med: Some nonlocal information can be accessed and used by each agent.

High: A wide range of nonlocal information, possibly in some abstracted form and including current, dynamic information, can be used by an agent for developing a partial global view.

4.2 Is predictive or constraining information present/developed for the domain?

Low: No, subproblems and subtasks do not interact in any meaningful way, or the influences are known beforehand, and the system is hardcoded to handle them.

Med: Yes. The agents use such information as is available beforehand to help them act in a coherent manner with one another as a total system.

High: Yes. The agents can also develop this information dynamically as it occurs in any particular problem.

4.3 Do the priorities of goals or tasks change dynamically?

Low: There are no relative priorities of goals or tasks.

Med: The relative priorities are known beforehand, or are assigned by some organizational or other fixed procedural means.

High: Priorities change dynamically, and the system responds to those changes.

4.4 How does an agent decide if a solution to a subproblem is globally consistent?

Low: It doesn't try to; all solutions to subproblems are globally consistent.

Med: Solutions are constructed in one central location and a human or oracle is consulted.

High: The consistency of a solution is contingent on how it fits in with other solutions, the problem data, etc. Agents must look at the consistency of their solutions with these things; there is no central arbiter of acceptability.

4.5 How does the system deal with an overconstrained set of goals?

Low: This situation never occurs.

Med: There is a central arbiter that decides how to relax constraints, or there is a local algorithm that guarantees an acceptable, globally consistent solution, or there is a common procedure for prioritizing goals to achieve the most important as a subset.

High: There is no arbiter. Constraints are relaxed by mutual agreement. Agents may decide (perhaps negotiate) on precisely what level constraint relaxation should take place dynamically, as well as who should be involved in such a decision.

4.6 Does any agent need to develop a partial global view?

Low: Agents develop only a local view of the problem.

Med: Some or all agents must develop a complete, global view.

High: Agents must develop a partial global (non-local) view, but not a complete global view.

5 Goal: Theories of Organizational Behavior and Control

Another goal of distributed AI is that of Coordination Theory [8] — how to effectively coordinate the activities of groups of computer and/or human actors. How do we organize such systems, especially if they are large? How do we control these systems? How do we reason about the control of the domain problem solving (if this is necessary)? How are problems broken down into simpler subproblems and solutions to these subproblems synthesized? More sophisticated systems are assumed to allow dynamic organizations and to reason on levels above the domain level (for example, reasoning about control).

5.1 Are multiple agents capable of solving a subproblem?

Low: No. There are specific agents for each subproblem.

Med: Yes and No. Some subproblems need special agents, but some don't, or multiple agents could have been used to solve a particular subproblem but were not.

High: Yes, multiple agents are capable and are used to solve a single subproblem.

5.2 How are tasks decomposed?

Low: Task decomposition is fixed at compile-time or system start-up, or tasks are inherently decomposed in the problem domain.

Med: Tasks are decomposed in a central location, or task decomposition is linked to task allocation (recursively).

High: A distinct, more complex, decentralized decomposition mechanism that is separate from the task allocation mechanism is used.

5.3 How are tasks allocated?

Low: Task allocation is fixed at compile-time or system start-up, or tasks are inherently allocated in the problem domain.

Med: Task allocation occurs using an organizational structure, or some other method where task allocation is fixed at the start of each individual problem.

High: Task allocation is dynamically opportunistic based on the availability of resources; a mixture of focused and open allocation.

5.4 How are results collected?

Low: There is no need to collect results, or result collection is the inverse of task allocation (it uses the same relationships).

Med: Result collection is independent of (and separate from) task allocation.

High: Result collection is independent and dynamically opportunistic (decided on during problem solving).

5.5 How autonomous is each agent in its actions?

Low: Agents have no autonomy whatsoever.

Med: Each agent is data directed and builds all of its goals locally (an "entrepreneurial" agent), or is goal directed and receives its goals from other agents (a "company" agent).

High: Each agent balances received and local goals in some reasoned manner.

5.6 How is network control distributed?

Low: It is not.

Med: In some fixed manner.

High: There is a control problem solving level that makes control distribution decisions.

5.7 How are network control problems resolved?

Low: They aren't resolved or there aren't any network control problems.

Med: By a central or otherwise known arbiter.

High: Through negotiation, or other mutual decision procedure among agents.

5.8 How are network control and domain problem solving related?

Low: There is no control problem solving.

Med: Control problem solving is tightly integrated with domain problem solving.

High: Control problem solving is done asynchronously from domain problem solving, possibly with a separate organizational structure.

5.9 What kind of organization is used?

Low: Master/Slave relationships between agents.

Med: A hierarchy or team.

High: Other, more complex organization (matrix?).

5.10 Can the organization or network control strategy evolve?

Low: No.

Med: It is fixed at the start of each problem.

High: It can change dynamically and opportunistically during problem solving.

5.11 How is network control related to domain control?

Low: No changeable local control mechanism.

Med: Network control is not strongly related to domain control.

High: Network control can effect changes and act in a synergistic fashion with local domain control (high in network control sophistication but low in domain independence).

5.12 Is there a network meta-level control component?

Low: No, or decisions are made at the start of a problem and are fixed thereafter.

Med: It allows fine tuning of network control.

High: It can cause a wide range of changes in network control dynamically and opportunistically during problem solving, and it is an integral part of the network control strategy.

5.13 Is predictive or constraining information present/developed for control?

Low: Not present.

Med: Present but not used.

High: The information is present and affects the control of the system.

5.14 What other information is used for network control?

Low: None.

Med: Any of: information about individual tasks, organizational information, agent goals, or agent plans for accomplishing goals.

High: Several of the above.

5.15 What assumptions are made about the quality of network control information?

Low: It is assumed consistent and complete.

Med: Information is assumed to be consistent, but possibly incomplete, or it is marked with belief or other uncertainty measures.

High: No assumptions are made about incoming information (i.e., it can be inconsistent, incomplete, and uncertain).

5.16 Are there special mechanisms to handle large numbers of agents?

Low: No.

Med: The organizational structure of agents (limits communication).

High: Another, more complex mechanism, such as network control based on an abstracted view of groups of agents.

5.17 Has the network control mechanism been empirically evaluated?

Low: No we have not, or no network control mechanism is used.

Med: Evaluation will follow implementation, or is just beginning.

High: Yes, many experiments have been conducted and results have been published.

6 Goal: Guaranteed Responsiveness and Fault Tolerance

Another goal of distributed AI research is to use multiple agents to attack specifically the problems of real-time domains and fault tolerance. Many claims have been made that multiple agents can help in both of these problems. Several questions about hardware appear under later goals but also apply to this one. A system receives a higher rating if it includes facilities for problem solving in real-time (time-constrained) domains or for solving problems in spite of multiple kinds of failures.

6.1 Does the system assume that fault tolerance is important?

Low: No, it is not important.

Med: It is a desirable feature, but not strictly necessary.

High: It is a requirement, or a necessary feature because of our environment or domain.

6.2 Do you assume that multiple kinds of failures are possible?

[Possible types of failures: processors, memory, sensors, effectors, external resources, communications, software.]

Low: None.

Med: Several.

High: All.

6.3 Do you assume that redundant features are available?

[Possible redundant features: processors, memory, sensors, effectors, external resources, communications, problem solving knowledge.]

Low: None.

Med: Several.

High: All.

6.4 Do agents generate deadlines for tasks?

Low: No.

Med: Yes, at the start of a task.

High: Yes, deadlines can be introduced at any stage of processing.

6.5 Is predictive timing information available?

Low: All task times can be predicted accurately, or no time information is available.

Med: Timings are known or computed at various levels of accuracy at the start of problem solving.

High: Timing information is dynamically computed and adjusted during problem solving.

6.6 How do agents respond to task deadlines?

Low: They ignore or don't have them, or plan for them individually in the short-term.

Med: They have several means available, such as changing the priorities of tasks, reassigning them to different agents, etc.

High: Agents may modify the scope or extent of tasks in order to get them done on time, or do extra processing if they have extra time.

6.7 How are faults detected and handled?

Low: They aren't handled at all.

Med: Faults can always be detected and another processor assigned, or tasks are simply assigned redundantly to agents.

High: Agents communicate with one another about overlapping areas of responsibility in order to detect faults or decide to redundantly solve problems in different ways in order to check consistency of results.

7 Goal: Effective CDPS Communications Protocols

One research goal is how distributed agents may effectively communicate, even when their information may not be complete, consistent, or certain. How will agents talk to one another if they do not share common knowledge representations? How do agents find out who there is to talk to? Higher ratings are given for more complex and less constrained or pre-specified communication protocols.

7.1 Do agents communicate functionally accurate information?

[Functionally accurate communication: agents communicate tentative results which may be incomplete, incorrect, or inconsistent with results produced by other agents [6]]

Low: No. Only completely accurate information is communicated.

Med: Tentative solutions may be transmitted between agents, but nothing else.

High: Agents have some representation to transmit tentative solutions, predictive information, belief or confidence in that information, etc.

7.2 How syntactically and semantically rich is the communication language?

Low: Object-oriented message passing (context-directed procedure invocation) or other fixed low-level language.

Med: Fixed high-level domain language, possibly with domain and control concepts.

High: Dynamically expandable high-level language with domain and control concepts or natural language.

7.3 What information do agents use to determine the when, whom, and what of communication?

Low: Long-term, fixed knowledge.

Med: A combination of long-term, fixed knowledge and local knowledge developed during problem solving.

High: Non-local knowledge received from other agents (such as goals, tasks, or problem solving state), local knowledge, and long-term, fixed knowledge.

7.4 How do agents with different structured information communicate?

Low: Does not occur — all information is structured the same.

Med: Agents communicate such information without any special reasoning, or specifically avoid communicating such information.

High: Communications include information about structure to actively allow reasoning about such information.

7.5 How are the paths of communication specified?

Low: Agent communication paths are predetermined. For example, objects in an object-oriented language communicate by messages.

Med: Agents know who they may communicate with beforehand, but the order and contents of that communication are not pre-specified.

High: Agent communication pathways are discovered as needed and used at the discrimination of the agent.

7.6 How are messages addressed?

Low: Broadcast to all agents (creates a scaling problem).

Med: Broadcast to small local groups, packet-switched, addressed, or point-to-point communications.

High: A full range of methods is available to use depending on the current situation.

8 Goal: Sophisticated Agents

Another research goal is to develop sophisticated agents to fill roles in a distributed system. What special properties or processes must an 'intelligent' agent have in order to be part of a CDPS system? More "sophisticated" agents are those that extensively model other agents in the system.

8.1 What *a priori* knowledge do agents have about other agents?

Low: None, and they do not develop any.

Med: They know which agents have what domain knowledge.

High: They have models of the beliefs, actions, plans, and reasoning processes of other agents.

8.2 What knowledge do agents develop about other agents?

Low: None.

Med: They learn which agents have what domain knowledge, or notice changes in this or the organizational structure.

High: They model the beliefs, actions, plans, and reasoning processes of other agents.

8.3 Can agents deal with inconsistent, incomplete, and uncertain information?

Low: No, agents assume that they have consistent, complete, and certain information.

Med: Agents do the best they can with the available data, and may use a simple uncertainty system.

High: Agents have a complex belief system that can represent uncertainty, lack of knowledge, negative belief, inconsistency, second order uncertainty, supporting and refuting evidence, etc.

8.4 Do agents employ tacit bargaining and detect hidden agendas?

Low: No.

Med: They do employ such tactics as a manner of course, but they have no reasoning process for deciding when to apply them.

High: They choose to employ these and other complex actions when appropriate.

9 Goal: System and Hardware Support

One of the major research goals is to build software infrastructures, in terms of languages, environments, and operating systems, that support CDPS. The same applies to hardware infrastructure. Higher ratings are given to those systems that have better specialized system support in those areas. Note that this can lead to domain and environmental assumptions, but it is perfectly acceptable for a system to grow in software and hardware support sophistication while not becoming sophisticated in the sense of handling many different domains.

9.1 What happens when a single hardware failure occurs?

Low: System crashes.

Med: If the system notices what is occurring, or in some other limited cases, the system remains operational.

High: System always remains operational and reports the failure, but may work with decreased efficiency until the failure is fixed.

9.2 What is the likelihood of a hardware failure?

Low: Hardware failure seldom occurs during problem solving and is not of concern.

Med: Failures occur frequently enough that the issue must be addressed.

High: Multiple faults can occur frequently.

9.3 How do computational resources affect problem solving?

Low: They don't.

Med: Availability of resources will affect efficiency of the CDPS system.

High: System operates in real time, and computational resources must be used effectively and efficiently to generate solutions in time.

9.4 What special resources does the environment provide?

Low: No environment.

Med: One or two of: domain problem solving expertise, debugging, tracing, or explanation support.

High: All of the above, including a language for building CDPS networks.

9.5 What special resources do the operating system or hardware provide?

Low: Nothing special; just network communications protocols, etc.

Med: Distributed operating system, including protection and access mechanisms, load balancing.

High: Operating system provides access to load balancing, estimated process times, etc., for use by the CDPS system (high on the sophisticated system support scale but low on environment independence).

9.6 Are the goals, tasks, priorities, etc. of an agent used by the underlying system?

Low: Most of these things are procedurally represented.

Med: There are explicit, easily accessed representations of these things, but not really used by the operating system.

High: They are explicitly articulated and accessed or used by the operating system (high in sophisticated system support but low in environmental independence).

9.7 How far along is the implementation?

Low: Research proposal to paper design.

Med: Prototype to working research system.

High: System is in real-world everyday use.

10 Goal: Develop general and representative hard domain problems

Not only do we wish to understand exactly what assumptions we make about our domains and environments, but we also want to find truly hard problems that exhibit the usefulness and necessity of CDPS techniques. A problem gets a higher rating if it leads to a system that has one of the other goals mentioned earlier — the need for cooperation, the need to acquire non-local information in order to make informed local decisions, the possession of real-time or fault-tolerant characteristics, the use of multiple levels of reasoning, or the presence of complex communications.

10.1 What is the relative speed of communication versus the amount of local computation?

Low: Communication is extremely fast and cheap and does not result in a large amount of local computation.

Med: About an even tradeoff between communication and computation.

High: Communication is much more expensive, or cheap communication can result in a large amount of local computation. Perhaps security (or other) considerations preclude most communication, or force agents to communicate using a great deal of shared knowledge so that individual acts are hard to understand outside of the agents' mutual context.

10.2 Do all system tasks *have* to be completed?

Low: Yes.

Med: No, depending on the specific problem.

High: The problem requires a tradeoff to be reasoned about, between what tasks are completed and what is left out, depending on deadlines, priorities, etc.

10.3 Can this problem be solved in a centralized manner?

Low: Yes.

High: No.

10.4 Is there any advantage to having multiple agents solve the same subproblem?

Low: No.

Med: Yes, simple fault tolerance.

High: Yes, there are different ways to solve the problem that result in different quality solutions or the ability to double check results, etc.

10.5 Is there significant uncertainty involved?

Low: No.

Med: One or two of: the input data is incomplete, there is uncertainty in the reliability of the input observations or observers, there is uncertainty in the processes or heuristics used by agents, or uncertain information must be aggregated or summarized from multiple sources of differing reliabilities.

High: There are multiple sources of uncertainty.

10.6 Are over-constrained situations (in terms of unachievable domain goals) possible?

Low: No.

Med: Rarely. It is not a primary consideration.

High: Often — it is a feature of the problem.

10.7 How easy is it to find a solution to the problem?

Low: Easy. The search space is dense with solutions, or the search space is small.

Med: There are multiple acceptable solutions, but also many wrong ends.

High: Hard. The search space is large and solutions few or unique, or there are multiple solutions in a large search space, but they range in their acceptability.

10.8 How much empirical evaluation has been done of the domain problem solving capabilities?

Low: None.

Med: Evaluation is being performed in conjunction with implementation.

High: Extensive testing, in conjunction with domain experts or others, has been performed and the results published.

11 Example CDPS Systems

We have chosen to look at two well-known projects and a new research effort that we have just initiated. The first, CNET, is the contract net system developed by Davis and Smith for the vehicle monitoring domain [10, 1]. The second, PGP, is the partial global planning framework developed by Durfee and Lesser for the Distributed Vehicle Monitoring Testbed (DVMT) [7, 3]. The third (denoted here GPGP+) is our new research effort, which focuses on extending the PGP framework to other domains, especially real-time domains. We will briefly describe the purpose and basic characteristics of each of these systems, and then compare them using our questions.

11.1 Contract Nets

Consider a network of loosely-coupled agents with various resources. If one of these agents receives a large problem to solve, then it has two choices: it can apply its own resources and solve the problem as best it can by itself; or it can decompose the large problem into smaller subproblems and convince other agents to pursue these subproblems. To make the best use of network resources, the agents should cooperatively solve large problems by assigning subproblems to suitable agents and working concurrently on these subproblems.

The contract-net protocol developed by Smith and Davis develops a framework for cooperating in this manner [1]. Given a task to perform, a node first determines whether it could break the task into subtasks that could be performed concurrently. If it forms such subtasks, or it is locally unable to perform the initial task, then the node must coordinate with others to decide where to transfer tasks. It employs the contract-net protocol to announce the tasks that could be transferred and request that nodes that could perform any of those tasks submit bids. A node that receives a task announcement message replies with a bid for that task, indicating how well it believes it can perform the task. The node that announced the task collects these bids and awards the task to the "best" bidder. The contract-net protocol allows nodes to broadcast bid-requests to all others or to focus bid-requests to a likely subset of nodes. Nodes can also communicate about their availability, so that focusing information (and decisions about whether it is worth advertising a task in the first place) can be based on dynamic views of the network.

The contract-net protocol promotes control based on negotiating over task assignments to form contractor-contractee relationships. These relationships determine how nodes will act and interact, and allow them to coordinate their activities to work together effectively. Because nodes exchange information about tasks and availability, they make dynamic control decisions about how they will cooperatively pursue tasks. Thus, in applications where the principal mode of interactions between nodes fits into the contracting model, the contract-net protocol is an effective approach for controlling cooperation [11]. However, there are applications which do not fit cleanly into this model of cooperation: tasks may be inherently distributed among nodes, and coordination is not a matter of decomposing and assigning tasks but instead is a matter of recognizing when distributed tasks (or partial results) are part of some larger overall task (or result) and, when this is the case, how to interact to achieve the larger task (or result).

11.2 Partial Global Planning

Partial global planning [3, 4] is a flexible approach to coordination that does not assume any particular distribution of subproblems, expertise, or other resources, but instead lets nodes coordinate in response to the current situation. Each node can represent and reason about the actions and interactions for groups of nodes and how they affect local activities. These representations are called **partial global plans** (PGPs) because they specify how different *parts* of the network *plan* to achieve more *global* goals. Each node can maintain its own set of PGPs that it may use independently and asynchronously to coordinate its activities.

A PGP contains an objective, a plan-activity-map, a solution-construction-graph and a status:

- The **objective** contains information about *why* the PGP exists, including its eventual goal (the larger solution being formed) and its importance (a priority rating or reasons for pursuing it).

- The **plan-activity-map** represents *what* the nodes are doing, including the major plan steps the nodes are concurrently taking, their costs and expected results, and why they are being taken in a particular order.

- The **solution-construction-graph** contains information about *how* the nodes should interact, including specifications about what partial results to exchange and when to exchange them.

- The **status** contains bookkeeping information for the PGP, including pointers to relevant information received from other nodes and when that information was received.

A PGP is a general structure for representing coordinated activity in terms of goals, actions, interactions and relationships.

When in operation, a node's PGPlanner scans its current network model (a node's representation of the goals, actions and plans of other nodes in the system) to identify when several nodes are working on goals that are pieces of some larger network goal (partial global goal). By combining information from its own plans and those of other nodes, a PGPlanner builds PGPs to achieve the partial global goals. A PGPlanner forms a plan-activity-map from the separate plans by interleaving the plans' major steps using the predictions about when those steps will take place. Thus, the plan-activity map represents concurrent node activities. To improve coordination, a PGPlanner reorders the activities in the plan-activity-map using expectations or predictions about their costs, results, and utilities. Rather than examining all possible orderings, a PGPlanner uses a hill-climbing procedure to *cheaply* find a better (though not always optimal) ordering. From the reordered plan-activity-map, a PGPlanner modifies the local plans to pursue their major plan steps in a more coordinated fashion. A PGPlanner also builds a solution-construction-graph that represents the interactions between nodes. By examining the plan-activity-map, a PGPlanner identifies when and where partial results should be exchanged in order for the nodes to integrate them into a complete solution, and this information is represented in the solution-construction-graph.

To control how they exchange and reason about their possibly different PGPs, nodes rely on a **meta-level organization** that specifies the coordination roles of each node. If organized one way, the nodes might depend on a single coordinator to form and distribute PGPs for the network, while if organized differently, the nodes might individually form PGPs using whatever information they have locally. The partial global planning framework lets nodes converge on common PGPs in a stable environment (where plans do not change because of new data, failed actions, or unexpected effects of their actions). However, when network, data and problem solving characteristics change and communication channels have delay and limited capacity, nodes can locally respond to new situations, cooperating effectively even when they have inconsistent PGPs.

11.3 Generalized Partial Global Planning and other extensions (GPGP+)

We have begun new research thrusts in generalized partial global planning and extensions (GPGP+) to handle the issues of real-time and meta-level control. A major focus of the proposed work is the extension and generalization of the partial global planning architecture that was developed for network control in, and is closely tied to, the specifics of the DVMT. We want to generalize this PGP network control architecture so it is non-DVMT-specific; thereby creating a generic protocol for control of CDPS that is applicable to a wider range of tasks. This involves developing and implementing a protocol and associated algorithms through which the user can tailor this protocol by the implementation of domain-specific code.

Generalized partial global planning is useful for ensuring globally coherent behavior in any CDPS system. Each agent is assumed to have a set of goals and subgoals that it intends to achieve. These goals may be related in many ways, such as being ordered in time or being alternate methods of achieving a single supergoal. For example, Agent $\mathcal{A}$ has a plan for a goal that has two subgoals, G5 and G9, that are being handled by agents $\mathcal{B}$ and $\mathcal{C}$. Agent $\mathcal{C}$, while working on goal G9 along with its own goals, generates a plan that includes goal G17, that can only be handled by agent $\mathcal{B}$. Agent $\mathcal{B}$ cannot act in a globally coherent manner unless it understands how its goals (G5 and G17) are related. It is these relations that generalized partial global planning is based.

11.3.1 Generic Relations

To handle these interactions, we are developing a set of generic relationships that will allow us to plan for goals. This fixed set of primitive relations will be domain independent, but for some relations the act of determining if that relation holds will be domain dependent. These primitives can be used to build generalized partial global plans in systems that have nothing to do with the DVMT. Using these relations, we will rework and extend the PGP algorithms to handle the coherent network control of CDPS systems other than the DVMT (like Pilot's Associate). A high-level protocol will allow an agent to specify and maintain its goal, plan, or task structure and transmit that structure to other agents for coordination purposes. Much like contract nets provide a domain-independent task allocation mechanism, we will build a generic network control system that will provide support for domain independent objects (such as agents, goals, plans, and tasks) and relations among those objects. The system builder provides his domain problem solving control and his own domain relationships, as well as support for the recognition of domain independent relationships. The generic PGP system then uses these relationships to bring about coherent network problem solving behavior, as well as providing a framework in which to build the system.

11.3.2 Real-time Control

The creation of a generic PGP that is appropriate for a wide class of applications will also involve extending the current PGP architecture to handle the issues of real-time control and large networks. From our perspective, real-time network control means not only scheduling to deadlines both periodic and non-periodic tasks, but also recognizing that for many applications there are trade-offs possible between the quality of the solution (e.g., certainty, precision, optimality) and the time needed to generate the solution. Often in real time situations planning is *reactive*, where the current situation mostly controls an agent's actions (where the "current situation" may include both local and global information), rather than *reflective*, where a sequence of actions is planned out in some detail before execution. This is because the agent must respond quickly, but more importantly, the agent may be too uncertain of the outcomes of its actions and of the changing world state to plan too far into the future. However, an intelligent agent will make use of periodic tasks, which occur in a predictable fashion, and known non-periodic tasks, to build a opportunistic planning framework that can keep an agent from painting itself into a corner with purely reactive planning techniques, or from exhaustively planning uncertain future details with reflective planning techniques.

11.3.3 Meta-level Control

In addition to extending the PGP protocol for real-time, it needs to be extended to effectively handle large networks of tens to hundreds of cooperating nodes. Experimentally, we have found it both unnecessary and computationally expensive to do detailed planning of all possible node interactions. For example, in a large, geographically distributed sensor interpretation system that covers the entire nation, it would rarely be necessary to perform detailed coordination between a sensor interpretation node in New York and one in California. However, it is important for regional centers, such as those developing maps for the East and Midwest, to coordinate at a gross level, balancing loads in areas of overlapping sensing and developing expectations of future loading based on current observations. There is also the need to develop closer interaction among nodes at the boundaries of regions based on the dynamics of the situation. We will initially be exploring an approach to solving this large network coordination problem by using an abstract description of the work loads of a group of nodes. We will extend the PGP framework to handle this more abstract description and introduce multiple levels of abstraction into the PGP framework.

Both of these extensions to the PGP framework will involve adding some form of meta-level network control. Further need for meta-level control arises because the full range of sophistication and responsiveness of the PGP architecture is not always required. There is a computational overhead for PGP control that is non-negligible; therefore, it should be used only in those situations where the benefits of sophisticated network control are warranted.

12 Comparisons

The previous examples can be used with the list of questions to show the state of some particular research from different viewpoints. The answer to each question denotes the position of a research project on some sophistication axis. Each project is placed at a low ($-$), medium ($\circ$), or high ($+$) sophistication. Our evaluations were assigned either on the basis of first-hand experience or from reading the literature — people with first-hand knowledge of a system might give different values for some questions.

Sometimes a project simply does not address the issue raised by a question. Such a system could have been marked low ($-$) because it does not deal with the issue, or high ($+$) because it does not preclude or prevent that issue from being solved, so anything is possible. Instead we use a fourth mark (?) to denote that the system in question does not deal with the issue, neither making restrictive assumptions nor providing sophisticated mechanisms. This mark can also be used when not enough information is available to answer the question.

Limit Domain and Environmental Assumptions		CNET	PGP/DVMT	GPGP+
2.1	How heterogeneous are the agents in your system?	+	○	+
2.2	What assumptions are made about the maximum number of agents in your system?	+	+	+
2.3	How is problem solving knowledge shared among agents?	?	○	?
2.4	How is short term knowledge acquired during problem solving replicated among agents?	?	+	+
2.5	How is consistency of short term knowledge maintained?	?	+	+
2.6	How is consistency of long term problem solving knowledge maintained?	−	−	−
2.7	Is the first solution obtained the correct one?	−	○	+
2.8	Do the agents make any assumptions about the operating system?	+	+	+
2.9	Can the agents interact with humans as agents?	?	?	?
2.10	Do you assume that new agents or communication paths can be dynamically added and deleted during problem solving?	+	+	+
2.11	Is problem solving continuous or discrete?	○	−	○
2.12	Has the system been evaluated in multiple domains?	+	−	−

Discover Paradigms for Building Cooperating Agents		CNET	PGP/DVMT	GPGP+
3.1	How do the goals of your agents interact?	−	○	+
3.2	Do you represent and reason about agent goals?	○	+	+
3.3	What range of collaboration is possible between agents?	?	○	+
3.4	How are domain conflicts resolved?	?	○	○
3.5	How do agents deal with non-reversible actions?	?	−	+
3.6	Can the prototypical problems (for which the system was designed) be solved without cooperation?	○	○	○

Develop Methods for Assuring Global Coherence	CNET	PGP/DVMT	GPGP+
4.1 What is the range of non-local information that agents use to assure coherent behavior?	$-$	$+$	$+$
4.2 Is predictive or constraining information present/developed for the domain?	$\circ$	$+$	$+$
4.3 Do the priorities of goals or tasks change dynamically?	$-$	$+$	$+$
4.4 How does an agent decide if a solution is globally acceptable?	$-$	$+$	$+$
4.5 How does the system deal with an overconstrained set of goals?	$-$	$\circ$	$+$
4.6 Does any agent need to develop a partial global view?	$+$	$+$	$+$

Theories of Organizational Behavior and Control	CNET	PGP/DVMT	GPGP+
5.1 Are multiple agents capable of solving a subproblem?	+	+	+
5.2 How are tasks decomposed?	○	+	+
5.3 How are tasks allocated?	+	+	+
5.4 How are results collected?	−	+	+
5.5 How autonomous is each agent in its actions?	○	+	+
5.6 How is network control distributed?	○	+	+
5.7 How are network control problems resolved?	−	+	+
5.8 How are network control and domain problem solving related?	○	+	+
5.9 What kind of organization is used?	○	+	+
5.10 Can the organization or network control strategy evolve?	−	○	+
5.11 How is network control related to domain control?	?	+	+
5.12 Is there a network meta-level control component?	−	−	+
5.13 Is predictive or constraining information present/developed for control?	○	+	+
5.14 What other information is used for network control?	−	+	+
5.15 What assumptions are made about the quality of network control information?	?	○	○
5.16 Are there special mechanisms to handle large numbers of agents?	−	○	+
5.17 Has the network control mechanism been empirically evaluated?	+	+	−

Guaranteed Responsiveness and Fault Tolerance	CNET	PGP/DVMT	GPGP+
6.1 Does the system assume that fault tolerance is important?	+	+	+
6.2 Do you assume that multiple kinds of failures are possible?	+	+	+
6.3 Do you assume that redundant features are available?	+	+	+
6.4 Do agents generate deadlines for tasks?	+	+	+
6.5 Is predictive timing information available?	−	+	+
6.6 How do agents respond to task deadlines?	○	○	+
6.7 How are faults detected and handled?	○	○	+

Effective CDPS Communications Protocols	CNET	PGP/DVMT	GPGP+
7.1 Do agents communicate functionally accurate information?	?	+	+
7.2 How syntactically and semantically rich is the communications language?	○	−	○
7.3 What information do agents use to determine the when, whom, and what of communication?	○	+	+
7.4 How do agents with different structured information communicate?	?	+	+
7.5 How are the paths of communication specified?	○	○	○
7.6 How are messages addressed?	+	+	+

Sophisticated Agents	CNET	PGP/DVMT	GPGP+
8.1 What a priori knowledge do agents have about other agents?	○	○	○
8.2 What knowledge do agents develop about other agents?	○	+	+
8.3 Can agents deal with inconsistent, incomplete, and uncertain information?	?	○	○
8.4 Do agents employ tacit bargaining and detect hidden agendas?	−	−	−

System and Hardware Support	CNET	PGP/DVMT	GPGP+
9.1 What happens when a single hardware failure occurs?	–	○	–
9.2 What is the likelihood of a hardware failure?	?	?	?
9.3 How do computational resources affect problem solving?	–	–	+
9.4 What special resources does the environment provide?	?	+	+
9.5 What special resources do the operating system or hardware provide?	–	–	–
9.6 Are the goals, tasks, priorities, etc. of an agent used by the underlying system?	–	–	?
9.7 How far along is the implementation?	○	○	–

Develop general and representative hard domain problems	CNET	PGP/DVMT	GPGP+
10.1 What is the relative speed of communication versus the amount of local computation?	+	+	?
10.2 Do all system tasks have to be completed?	–	○	+
10.3 Can this problem be solved in a centralized manner?	–	–	–
10.4 Is there any advantage to having multiple agents solve the same subproblem?	?	+	+
10.5 Is there significant uncertainty involved?	?	+	+
10.6 Are over-constrained situations possible?	–	–	○
10.7 How easy is it to find a solution to the problem?	?	○	○
10.8 How much empirical evaluation has been done of the domain problem solving capabilities of your system?	–	+	–

As was discussed earlier, each question may relate to more than one goal but appears only once in the set of questions. For example, Question 5.15 ("What assumptions are made about the quality of network control information?") applies to both Section 5 (Network Control) and Section 2 (Limiting Assumptions).

Sometimes, a question may have the opposite meaning for two goals. This is indicated in the table below by a bar over a bold number. For example, Question 9.6 ("Are the goals, tasks, priorities, etc. of an agent used by the underlying operating system?”) is both a question about system support and limiting assumptions, with a “high” answer to 9.6 implying high complexity in system support but low in limiting environmental assumptions.

The following table illustrates a mapping of questions to goals (other than the ones they appear with) that they influence:

Limit Assumptions	$\overline{\mathbf{5}.11}$,5.15,6.2,$\overline{\mathbf{9}.5}$,$\overline{\mathbf{9}.6}$,10.1
Cooperating Agents	2.1,2.3,2.5,2.6,4.5,5.1
Global Coherence	2.4,2.7,3.1,5.5,5.16,10.1
Network Control	2.2,3.3,10.1
Real Time & Faults	4.3,5.13,9.1,9.2,9.3
Communications	2.9,2.14,4.1,10.1
Sophisticated Agents	2.9,3.2,3.3,4.2,5.5
System Support	$\overline{\mathbf{2}.8}$,2.14,5.17,10.8
Domain Problem	2.7,2.16,3.1,3.6,4.2,9.2

Now for each goal we can add up all of the low, medium, and high answers for the questions in that section *and* the related questions in the table above, ignoring the (?). Those evaluations can then be summarized as follows.

	CNET				PGP/DVMT				GPGP+			
	$-$	$\circ$	$+$	?	$-$	$\circ$	$+$	?	$-$	$\circ$	$+$	?
Limit Assumptions	2	1	9	6	3	6	8	1	3	2	9	4
Cooperating Agents	3	2	2	5	3	6	3	0	1	2	8	1
Global Coherence	7	2	2	1	0	4	8	0	0	0	11	1
Network Control	6	6	5	3	1	4	15	0	1	1	17	1
Real Time & Faults	4	3	4	1	1	3	7	1	1	0	10	1
Communications	1	3	3	3	2	1	6	1	0	2	5	3
Sophisticated Agents	1	5	0	3	1	3	4	1	1	2	5	1
System Support	6	1	2	2	4	2	4	1	6	0	3	2
Domain Problem	6	2	2	4	3	5	5	1	3	3	6	2

Evaluations such as this can be used in two ways. First, we can use then to compare the strengths, weaknesses, and different research thrusts of different CDPS systems. Secondly, and more usefully, we can use them to look at how new research will make advances in the field over older projects.

We can summarize the information in the above table by counting a $(+)$ mark as a 1, a $(-)$ mark as a -1, a $(\circ)$ as a 0, and summing them. The table below shows the resulting values and the number of unanswered questions (?):

	CNET		PGP/DVMT		GPGP+	
	sum	?	sum	?	sum	?
Limit Assumptions	7	6	5	1	6	4
Cooperating Agents	-1	5	0	0	7	1
Global Coherence	-5	1	8	0	11	1
Network Control	-1	3	14	0	16	1
Real Time & Faults	0	1	6	1	9	1
Communications	2	3	4	1	5	3
Sophisticated Agents	-1	3	3	1	4	1
System Support	-4	2	0	1	-3	2
Domain Problem	-4	4	2	1	3	2

Looking at the table above as an aid to comparisons, we can see some large characteristics that point out the major differences between the evaluated systems:

- Although we felt that CNET made less assumptions about its domain than PGP, due to the uncertainty surrounding details of CNET this is only marginally apparent in the table.

- GPGP+ ranks significantly better than PGP or CNET on creating cooperating agents that can handle complex and sophisticated goal interactions, which is to be expected since one of the focuses of GPGP is to reason about the goals of agents so that they can cooperate more effectively.

- Both PGP and GPGP+ rank significantly higher than CNET in providing facilities for agents to act in a coherent manner, in creating sophisticated network control strategies, and in working in real-time and fault-tolerant situations.

- Little difference between the systems is seen in terms of communication protocols.

- PGP and GPGP+ rank marginally better than CNET in providing a sophisticated agent framework where agents reason about other agents explicitly. Perhaps more questions need to be aimed specifically at this area.

- PGP, as the most mature system presented, ranks higher than the others in system support and empirical evaluation.

- The domain problems of the PGP and GPGP+ frameworks show more promise than that of CNET, primarily because the vehicle monitoring problem that the DVMT solves[7] was extended to make it more complex than the problem described in the CNET paper[1].

The second way to use this data is to see how new work will extend the capabilities of the old. For example, besides the GPGP+ work described here, researchers are pursuing other work that will extend our abilities in building sophisticated CDPS systems. Work is being done on complex forms of negotiation and constraint relaxation between agents that will move us higher along the complexity axes in questions 3.4, 4.5, and 7.4. Other work on reasoning with uncertainty will move us farther along on the axes of questions 5.15, 7.1, 7.4, 8.3, and 10.5.

13 Conclusions

Cooperative distributed problem solving, like its parent field AI, is vaguely defined because it is precisely those problems that are ill-defined (admitting to no clear algorithmic solutions) that are of interest. Developing a set of questions for comparing and evaluating research projects is thus an ill-defined task in itself, where the lack of structure in the field leads to a sense of uncertainty about how to characterize the dimensions of the field. Ours is one of several attempts to chart these dimensions [5, 2, 9], and we feel it is the most complete to date for understanding and comparing similar projects, but as the field evolves, so will the questions. Indeed, many of the questions we ask in this paper were not and could not be considered in the early days of CDPS research when the field was even less defined.

Our hope is that our new research directions will serve to answer some questions and raise even more interesting new questions. As is clear from its description and comparisons among systems, our proposed work is intended to meet and often exceed the capabilities of past systems. Because it has yet to be implemented, however, we cannot realize the full implications of even more sophisticated cooperation among computers, in terms of their ability to achieve new results and to work more effectively with humans. Because our long term goal is to build computers that can intelligently work together and with us, we must continue to ask the questions put forth in this paper, and to extend this set of questions as we learn more about cooperative distributed problem solving.

Acknowledgment

We would like to thank Dr. Robert Simpson who inspired us to try to develop an approach to understanding how to measure progress in the field and discuss critically the scientific content of different research projects. We would also like to thank Dan Corkill, Dan Neiman, and the reviewers for their perceptive comments.

References

[1] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[2] Keith S. Decker. Distributed problem solving: A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(5):729–740, September 1987.

[3] 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*, August 1987.

[4] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

[5] M. N. Huhns, editor. *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.

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

[7] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed. *AI Magazine*, 4(3):63–109, Fall 1983.

[8] Thomas W. Malone. What is coordination theory? In *Proceedings of the National Science Foundation Coordination Theory Workshop*, February 1988.

[9] R. G. Smith. Report on the 1984 distributed artificial intelligence workshop. *AI Magazine*, 6(3):234–243, Fall 1985.

[10] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, January 1981.

[11] H. Van Dyke Parunak. Manufacturing experience with the contract net. In *Proceedings of the 1985 Distributed Artificial Intelligence Workshop*, pages 67–91, December 1985.

Contents