Trends in Cooperative Distributed Problem Solving

Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill

July 12, 1995

[1]

[1]Edmund H. Durfee is with the Department of Electrical and Computer Engineering at the University of Michigan. Victor R. Lesser and Daniel D. Corkill are with the Department of Computer and Information Science at the University of Massachusetts.

1 Introduction

Cooperative Distributed Problem-Solving (CDPS) studies how a loosely-coupled network of problem solvers can work together to solve problems that are beyond their individual capabilities. Each problem-solving node in the network is capable of sophisticated problem solving and can work independently, but the problems faced by the nodes cannot be completed without cooperation. Cooperation is necessary because no single node has sufficient expertise, resources, and information to solve a problem, and different nodes might have expertise for solving different parts of the problem. For example, if the problem is to design a house, one node might have expertise on the strength of structural materials, another on the space requirements for different types of rooms, another on plumbing, another on electrical wiring, and so on. Different nodes might have different resources: some might be very fast at computation, others might have connections that speed communication, while still others might have more memory. Finally, different nodes might have different information or viewpoints on a problem. For example, geographically separated nodes that are monitoring aircraft movements will have different perceptions because their sensors will pick up different signals. Only by combining information about their views will they be able to form an overall picture of aircraft movements.

CDPS nodes cooperatively solve a problem by using their local expertise, resources, and information to individually solve subproblems, and then integrating these subproblem solutions into an overall solution. As they work together, the nodes face two very important constraints. First, because their subproblem solutions must eventually be integrated, their individual solutions must fit into an overall solution. Even in situations where this overall solution is not represented at any one node, the distributed components of the solution must still be mutually consistent. Thus, the nodes must coordinate their asynchronous and parallel problem solving to build compatible solutions to their interdependent subproblems. The second constraint is that nodes are limited in how much they can communicate. Limited internode communication stems from either inherent bandwidth limitations of the communication medium or because of the high computational costs of packaging and assimilating information to be sent and received among nodes. Limited communication between nodes that are working on interdependent subproblems means that nodes must rely on sophisticated local reasoning to decide on appropriate actions and interactions. Each node must be capable of modifying its behavior as circumstances change and planning its own communication and cooperation strategies with other nodes.

1.1 Why CDPS?

From this description of CDPS, we might ask: If coordination among problem solvers is difficult, why not build a single, more powerful problem solver to perform the functions of a CDPS network? In short, why CDPS?

One answer to these questions has a technological basis. Advances in hardware technology for processor construction and interprocessor communication make it possible to connect together large numbers of powerful, yet inexpensive, processing units that execute asynchronously. Interconnected processors can be a cost-effective way to provide the computational cycles required by AI applications. A range of connection structures is possible, from a very tight coupling of processors through shared or distributed memory, to a looser coupling of processors through a local area communication network, to a very loose coupling of geographically distributed processors through a communication network. Regardless of how tightly they are coupled, the processors must use their communication medium selectively, otherwise they might overwhelm each other with more information than they can process. Whether because of limited communication bandwidth or limited processing power, the processors cannot share all of their information, but must be able to work effectively together anyway.

A second answer is that many AI applications are inherently distributed. Some are spatially distributed; for example, interpreting and integrating data from spatially distributed sensors or controlling a set of robots that work together on a factory floor. Other applications are functionally distributed, such as bringing together a number of specialized medical-diagnosis systems on a particularly difficult case or developing a sophisticated architectural expert system composed of individual experts in specialties such as structural engineering, electrical wiring, and room layout. Finally, some applications are temporally distributed (pipelined), as in a factory, where production lines consist of several work areas, each having an expert system responsible for scheduling orders. A CDPS network that manages the distribution of data, expertise, processing power, and other resources has significant advantages over a single, monolithic, centralized problem solver. These advantages include: faster problem solving by exploiting parallelism; decreased communication by transmitting only high-level partial solutions to nearby nodes rather than raw data to a central site; more flexibility by having problem solvers with different abilities dynamically team up to solve current problems; and increased reliability by allowing problem solvers to take on the responsibilities of problem solvers that fail.

A third answer stems from the principles of modular design and implementation. The ability to structure a complex problem into relatively self-contained processing modules leads to systems that are easier to build, debug, and maintain and that are more resilient to software and hardware errors than a single, monolithic module. For example, the general field of medical diagnosis is complicated and extensive. To manage the field, medical experts divide it (and themselves) into many specialties. If we wanted to build a general medical diagnosis system, we could exploit the modularity of the field, building knowledge-based systems for each specialty in parallel and with a minimum of interaction between systems. Because they are more focussed, debugging and maintaining these smaller systems would be much simpler than for a single, colossal system. For example, the individual systems could more easily maintain internally consistent knowledge when the overall knowledge base is inherently inconsistent. Clearly, this would be a route to implementing such comprehensive systems, provided that the underlying CDPS technology was in place so that the separate systems could work together as human specialists can.

A fourth answer has an epistemological basis. Cooperation, and more generally coordination, are complex and little understood phenomena. One approach to validating theories about such phenomena is to develop and test computer models that embody those theories. Just as AI systems are used to validate theories of problem solving and intelligence in linguistics, psychology, and philosophy, CDPS systems can help validate theories in sociology, management, and organizational theory. Moreover, the technology developed in these studies could lead to more effective use of computers as tools to improve coordination among people.

Finally, a fifth answer has a societal foundation. One of the goals of AI is to develop systems that become part of our everyday world. These systems would perform many of the mundane and boring tasks that require human intelligence, thereby freeing up human intelligence so that it can be reserved for more exciting meditations. To be accepted, these AI systems must interact with people on even terms—if people must become "AI literate" to use them, the systems will

1. Increase the task completion rate through parallelism.
2. Increase the set or scope of achievable tasks by sharing resources (information, expertise, physical devices, etc.).
3. Increase the likelihood of completing tasks (reliability) by undertaking duplicate tasks, possibly using different methods to perform those tasks.
4. Decrease the interference between tasks by avoiding harmful interactions.

Table 1: Generic Goals of Cooperation

not be accepted into human society. Therefore, AI systems must have the ability to flexibly and intelligently cooperate and coordinate with each other and with people. Although this long-term goal remains distant, CDPS represents steps toward it.

1.2 An Overview of CDPS Goals and Applications

CDPS, like other areas of AI, realizes theoretical advances in applications. In this section, we describe the general goals of CDPS, some tasks in which achieving these goals is important, and some specific technical problems that must be solved for CDPS to succeed. Four general goals for CDPS are presented in Table 1, [21]. Depending on the kind of application, some of these goals might be more important than others. The differences are highlighted by considering several classes of application domains in CDPS research.

Distributed Interpretation. Distributed interpretation applications require the integration and analysis of distributed data to generate a (potentially distributed) semantic model of the data. Application domains include distributed sensor networks [51, 50, 56, 89] and communication network fault diagnosis [12]. In these applications, a central problem solver is inappropriate because it would be less reliable than a network, it would not exploit the potential parallelism in processing data from different regions, and it would require substantial communication bandwidth to collect the large amounts of raw sensory data. A CDPS network is more reliable (performance degrades gracefully if individual nodes fail), would work on different parts of interpretations in parallel, and would communicate a relatively small number of high-level interpretations. However, to realize these benefits, the individual problem solvers must each form partial solutions that are globally relevant. The nodes must selectively exchange enough information to allow each to decide what data to interpret in order to form such partial solutions. They also must coordinate the selective exchange of partial interpretations so that they can help each other resolve ambiguities and can integrate local results into complete interpretations.

Distributed Planning and Control. Distributed planning and control applications involve developing and coordinating the actions of distributed effector nodes to perform desired tasks. Application domains include distributed air-traffic control [23, 88], cooperating robots, remotely piloted vehicles [82], distributed process control in manufacturing [80, 62, 61], and resource allocation/control in a long-haul communication network [11, 34]. Distributed planning and control applications often involve distributed interpretation to determine appropriate node actions. As in the distributed interpretation domains, data is inherently distributed among nodes, in this case because each has its own local planning database, capabilities, and view of the world state. Moreover, the combinatorics of planning potential activities for a large number of nodes working in parallel can be overwhelming. Even if it were possible, reliance on a central controller would generally lead to a slow system (because of communication delays between controller and controllees), and it would not be robust. A CDPS approach reduces these problems, but once again, because no single node has an overall view of the network, nodes must work together to coordinate their actions and interactions.

Cooperating Expert Systems. One means of scaling expert-system technology to more complex and encompassing problem domains is to develop cooperative interaction mechanisms that allow multiple experts systems to work together to solve a common problem. Illustrative situations include controlling an autonomous vehicle that uses separate expert systems for system status, mission planning, navigation, situation assessment, and piloting [3, 74]; or negotiation among expert systems of two corporations to decide price and/or delivery time on a major purchase. The heterogeneous character of cooperating experts means that individual agents might have different goals, different approaches to problem solving, different evaluation criteria for solutions, and different internal representations of problems and solutions. Getting them to cooperate is not simply a matter of giving them a common communication language. For them to reconcile different solutions to the same problem, they need detailed models of each other—their capabilities, goals, plans, and preferences—to form a compromise solution, or to re-evaluate their solution criteria to cooperatively generate a completely new solution [46].

Computer-Supported Human Cooperation. Computer technology promises to provide people with more and better information for making decisions. However, unless computers also assist people by filtering the information and focusing attention on relevant information, the amount of information can become overwhelming [8, 41, 54]. By building AI systems with coordination knowledge, we can remove some of the burden from people. Domains where this is important include intelligent command and control systems and multiuser project coordination [16, 57, 60, 70]. By building networks of CDPS computer assistants for the people in an organization, we can improve coordination by allowing these assistants to solve (initially routine) coordination problems such as scheduling meetings or routing messages to suitable people. The advantages of this CDPS approach, besides releasing humans from many coordination tasks, are that the assistants can work in parallel, can dynamically team up to coordinate the people they assist, and can work behind the scenes (in the background and at night) to share relevant information for making more coordinated decisions.

Cognitive Models of Cooperation. Although the designers of CDPS approaches have consistently used insights about human cooperation to build similar capabilities into their systems, little research to date has worked in the opposite direction. But AI methods have in the past served to implement and validate theoretical models of human intelligence, and CDPS provides a similar methodological framework for testing theories about human cooperation and coordination. For example, an important and common aspect of coordination among humans is negotiating through compromise [85]. Developing mechanisms that emulate human methods for coordinating their interactions can improve our understanding of humans, and in particular about how humans iteratively converge on decisions about how to share resources and avoid detrimental interactions.

These application areas provide context for refining the general goals of cooperation (Table 1) into specific goals. We can list several specific goals (where numbers in parentheses refer to corresponding generic goals), and indicate one of possibly many application domains where they arise:

- Increase the solution creation rate by forming subsolutions in parallel (1). For example, distributed interpretation and distributed planning involve building partial interpretations or partial plans in parallel.
- Minimize the time nodes must wait for results from each other by coordinating activity (1,4). For example, in distributed interpretation the nodes should coordinate how, when, and where to exchange local interpretations to quickly construct an overall interpretation.
- Improve the overall problem solving by permitting nodes to exchange predictive information (2). For example, an expert system for room layout might quickly generate and share a rough view of a kitchen, so that the expert systems for wiring and plumbing can better predict where in the room they should concentrate their efforts.
- Increase the probability that a solution will be found despite node failures by assigning important tasks to multiple nodes (3). For example, in distributed planning the nodes might be organized into small groups, each with a leader to coordinate the group. But the leader's information and expertise should also be available in one or more of the group members just in case the leader is unavailable due to hardware or communication failures.
- Improve the use of physical resources by allowing nodes to exchange tasks (2). For example, in computer-supported human cooperation, some assistants might be better suited for mail routing (have better network connections) while others might be better at scheduling meetings (have more computing capability for exploring possible schedules). Assistants should be able to exchange tasks to take advantage of each other's strengths.
- Improve the use of individual node expertise by allowing nodes to exchange goals, constraints, partial solutions, and knowledge (programs) (2). For example, cooperating expert systems should exchange information about constraints to focus how each applies its expertise to the current problem.
- Reduce the amount of unnecessary duplication of effort by letting nodes recognize and avoid useless redundant activities (4). For example, in distributed interpretation, nodes with overlapping sensors and identical expertise should not interpret data about the same phenomena if their interpretations are likely to be redundant.
- Increase the confidence of a (sub)solution by having nodes verify each other's results through rederivation using their individual expertise and information (2,3). For example, in distributed interpretation, nodes with overlapping sensors but different expertise might each interpret their shared data to make sure they get consistent results.

- Increase the variety of solutions by allowing nodes to form local solutions without being overly influenced by other nodes (1,4). For example, in distributed interpretation each node should not necessarily ignore important local information just because it is incompatible with preferred network interpretations—the network might be wrong!

- Reduce the communication resource usage by being more selective about what messages are exchanged (4). For example, in modeling human negotiation through compromise, the nodes should not exchange all the details about how they came up with a proposal, but instead should exchange just enough information to converge on an agreeable compromise.

These specific goals suggest that maximizing one specific dimension of effective cooperation, such as speed of solution, makes it impossible to achieve effective cooperation along other dimensions, such as limiting internode communication, or high reliability in case of hardware failure. Thus, effective CDPS network control involves balancing efficient use of communication and processing resources, high reliability, responsiveness to unanticipated situations, and solution quality based on application-specific criteria. The emphasis is shifted from optimizing a specific dimension of effective cooperation to achieving a balance that leads to acceptable network performance. This approach is similar to the concept of "satisficing" developed by March and Simon to describe human organizational problem solving [55].

Network coordination is difficult in CDPS networks because limited internode communication restricts each node's view of network problem solving activity. Network control must be able to tolerate the lack of up-to-date, incomplete, or incorrect control information due to delays in the receipt of information, the high cost of acquisition and processing of the information, and errors in communication and processing hardware. Furthermore, it is important that network coordination policies do not consume more processing and communication resources than are saved by the increased problem solving coherence. Thus, the cooperative problem solving necessary for effective network control may itself require a satisficing approach. Corkill and Lesser suggest that even in networks composed of a modest number of nodes, a complete analysis to determine the detailed activities at each node is impractical; the computation and communication costs of determining the optimal set and allocation of activities far outweigh the improvement in problem-solving performance [14]. Instead, they suggest that coordination in CDPS networks must sacrifice some potential improvement for a less complex coordination problem. Problem solving and coordination must be balanced so that the combined cost of both is acceptable.

1.3 What CDPS Is Not

Networks of cooperating nodes, both conceptual and actual, are not new to AI. However, the relative autonomy and adaptability of the problem-solving nodes—a direct consequence of limited communication—sets CDPS networks apart from approaches such as the Actor framework [38], Hearsay-II [22], the ETHER language [44], the BEINGS system [48], CAOS [71], Poligon [64], and Connectionism [58]. In all of these systems, knowledge is compartmentalized so that each actor or "expert" is a specialist in one particular aspect of the overall problem-solving task. The cooperative behavior exhibited by these systems stems from either a centralized scheduling mechanism or from predefined interactions between tightly-coupled, simple processing elements. Each "expert" has little or no knowledge of the problem-solving task as a whole or of general strategies for communication and cooperation. As a result, an expert cannot function outside the context of the other experts in the system nor outside specific communication and cooperation protocols specified in advance by the system designer. Metaphorically speaking, an element in these networks is like a subsystem of the brain, because intelligence emerges from a well-structured and tightly-connected collection of such elements.

In contrast, each node in a CDPS network possesses sufficient overall problem-solving knowledge that its particular expertise (resulting from a unique perspective of the problem-solving situation) can be applied and communicated without assistance from other nodes in the network. This does not imply that a node functions as well alone as when cooperating with other nodes—internode cooperation is often the only way of developing an acceptable overall solution—but every node can at least formulate a partial solution using only its own knowledge. Each node also possesses significant expertise in communication and control strategies. This knowledge frees the network from the bounds of designed protocols and allows nodes the flexibility to develop their own communication and cooperation strategies dynamically. Appealing to a different metaphor, an element in a CDPS network is like a person who is part of a team: each person can perform well with minimal intervention from teammates, but the team as a whole performs well only when the people coordinate their individual actions.

CDPS is a subfield of the larger field of Distributed AI. Distributed AI studies intelligent coordination: How do intelligent systems make decisions that allow them to achieve their goals in a world populated by other intelligent systems with *their* own goals? When deciding on an action to take, an intelligent system should use whatever knowledge it has to consider the potential actions of other intelligent systems and the effects of those actions. Reasoning about other systems is important whether the systems are cooperating, coexisting, or competing with each other. CDPS, as its name implies, concentrates only on forms of cooperation between AI systems. Note, however, that cooperation does not assume *benevolence* between nodes [66], but instead that despite different viewpoints and goals, nodes must work together to meet the demands of their environment.

CDPS also differs significantly from distributed processing. A distributed-processing network typically has multiple, disparate tasks executing concurrently in the network. 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 function. However, more recent research into distributed scheduling for a network operating system has begun to take more of a CDPS perspective [81].

2 Important CDPS Approaches and Empirical Investigations

The underlying problem encountered in CDPS networks is that nodes must make decisions about their problem-solving and communication actions based on local views that might be incomplete, inconsistent, or out-of-date. They must use their communication and computation resources to not only perform in the application domain (domain problem solving), but also to control their actions and interactions. Solving this **control problem** entails dealing with interactions among subproblems (in the application domain), promoting parallelism, integrating local results into complete solutions, resolving inconsistencies among local results, and transferring relevant information at appropriate times.

Almost every CDPS approach developed to date has been motivated and evaluated [17] in the context of an application domain, often by building a simulator for the domain . Here, we discuss major CDPS approaches in the context of these applications to highlight the CDPS issues that the applications exemplify and to describe empirical evidence (from simulations) for evaluating the capabilities and the costs of the approaches. In the following discussion, it is important to remember that the implementations are prototypes and simulations; to date, no CDPS networks have actually been used in real-world applications.

Important CDPS approaches can be categorized in terms of:

Negotiation: Using dialogue among nodes to resolve inconsistent views and to reach agreement on how they should work together to cooperate effectively.

Functionally-Accurate Cooperation: Overcoming inconsistency by exchanging tentative results to resolve errors and converge on problem solutions.

Organizational Structuring: Using common knowledge about general problem-solving roles and communication patterns to reduce nodes' uncertainty about how they should cooperate.

Multi-Agent Planning: Sharing information to build a plan for how agents should work together, then distributing and following this plan throughout problem solving.

Sophisticated Local Control: Integrating reasoning about other agents' actions and beliefs with reasoning about local problem solving, so that coordination decisions are part of local decisions rather than a separate layer above local problem solving.

Theoretical Frameworks: Using mathematical and logical models of agents, their beliefs, and their reasoning to understand the theoretical capabilities of CDPS networks.

With a few exceptions, we will not discuss centralized approaches—where the network has a single coordinating node—because centralized control is contrary to CDPS. That is, CDPS focuses on networks and problem applications where centralization is not a viable option, for reasons such as limited computation (the problem of coordinating many nodes is computationally intractable for a single coordinator), limited communication (a single coordinator could be a communication bottleneck and could be overwhelmed with information from the network), and reliability (the network performance should not rely on one node).

2.1 Negotiation

Negotiation is a fundamental part of human cooperation, allowing people to resolve conflicts that could interfere with cooperative behavior. A perennial goal of CDPS researchers has been to capitalize on insights about human negotiation, to build mechanisms that enable AI systems to negotiate.

Unfortunately, negotiation, like other terms that describe human behavior (e.g., "intelligence") is difficult to define in mechanistic terms. For example, Sycara [86] states that

> the negotiation process involves identifying potential interactions either through communication or by reasoning about the current states and intentions of other agents in the system and modifying the intentions of these agents to avoid harmful interactions or create cooperative situations.

We define negotiation as *the process of improving agreement (reducing inconsistency and uncertainty) on common viewpoints or plans through the structured exchange of relevant information.* Although these descriptions of negotiation capture many of our intuitions about human negotiation, they are too vague to provide blueprints for how to get AI systems to negotiate. The following subsections discuss more specific characterizations of negotiation.

2.1.1 The Contract-Net Protocol

One of the earliest and most influential research projects in CDPS is the Contract-Net framework developed by Smith and Davis [79, 75, 76, 77, 78]. The nodes use the Contract-Net protocol to form contracts concerning how they should allocate tasks in the network. **Contracting** involves an exchange of information between interested parties, an evaluation of the information by each member from its own perspective, and a final agreement by mutual selection. It differs from **voting** in that dissident members are free to exit the process rather than being bound by the decision of the majority.

In the Contract-Net protocol, nodes coordinate their activities through contracts to accomplish specific goals. Contracts are elaborated in a top-down manner. At each stage, a **manager** node decomposes its contracts into subcontracts to be accomplished by other **contractor** nodes. This process involves a bidding protocol based on a two-way transfer of information to establish the nature of the subcontracts, and to determine which node will perform a particular subcontract. The elaboration procedure continues until a node can complete a contract without assistance. The result of the contract elaboration process is a network of control relationships, in the form of manager/contractor relationships, distributed throughout the network.

Before contracting can begin, a manager must recognize that it has a task to be allocated. Where this task comes from depends on the application and the manager's reasoning methods. Typically, the manager receives a large task, and decomposes it into smaller tasks in a predefined way. It announces a task to the network, and nodes that are currently idle receive and evaluate the announcement. Nodes with the appropriate resources, expertise, and information reply to the manager with bids that indicate their suitability to perform the task. After sufficient time has elapsed, the manager evaluates the bids it has received, and awards the task to the most suitable node, or to several suitable nodes if redundancy is needed to ensure reliability. Finally, manager and contractor focus message exchanges between themselves: the manager supplies task information and the contractor reports progress and, hopefully, the eventual result of the task.

Smith and Davis investigated the performance of the Contract-Net protocol in several application domains. For example, they investigated a distributed interpretation application, where the network should track vehicles over a large geographical area. Their spatially distributed network was composed of two types of nodes: sensor nodes that can extract signal features from the data they sense, and manager nodes that can process the signal features from several sensor nodes to construct a map of vehicle movements. A manager node desires to form contracts with sensor nodes that are adequately distributed around an area and that have a complement of sensory capabilities.

On the other hand, a sensor node wants to interact with a nearby manager to minimize communication. The Contract-Net protocol allows manager and sensor nodes to each have input into the contracts that are formed.

This application illustrates the use of message structures in the Contract-Net protocol, depicted in Figure 1. Every message includes information about its source, destination, type, and contract identifier. A task announcement message includes abstract information about the task, expected capabilities of potential contractors, the information that a bid should contain, and a deadline for when bids should be received. In the vehicle monitoring application, the task abstraction specifies the task type and the manager's location; the expected capabilities indicate that a contractor must have certain sensory abilities and be in a particular area; and the information a bid should contain includes the sensor's location and sensory abilities. Furthermore, in the case of a task announcement, the message destination could indicate that the announcement should be broadcast to every node, or, if a manager node has information about which nodes are appropriate for the task, it can use *focussed addressing* to send the message only to them.

Upon receipt of a task announcement, a node may send a task bid to the manager that announced the task. Besides the source, destination, type, and contract identifier, a task bid message includes the information requested in the task announcement's bid specification. In the vehicle monitoring application, the bid indicates the position and sensory capabilities of the sensor node.

Finally, following expiration of the task announcement, the manager evaluates the bids and builds a task award message for each node that is awarded the task. In the vehicle monitoring application, the task award message indicates which of a sensor node's sensory capabilities are requested by the manager.

In addition to the basic message types, the Contract-Net protocol also allows nodes to announce their availability; this provides dynamic information to managers about loading in the network so that a manager can use focussed addressing to reduce network communication. Other features of the protocol include special cases in which a contractor can avoid bidding and award a contract directly.

The Contract-Net protocol provides common message formats and a shared communication structure (i.e., nodes know about the order of message exchange to generate contracts). However, to use this protocol effectively, a manager must use knowledge about the particular application to decide which tasks to contract out, how to describe an announced task, where to send the announcement, and how to select the best contractor(s). In turn, a contractor must be able to interpret the task needs from the message and to decide whether and how to bid on the task.

As a general framework for exchanging messages, the Contract-Net protocol does not include the application-dependent knowledge that nodes need to make these decisions. Hence, it does not prescribe how nodes should cooperate. But it does provide a language for nodes to use when exchanging information that can lead to these decisions. In addressing the issue of how to structure communication to allow contracting, the Contract-Net protocol answered some questions but raised many others that CDPS researchers are still trying to answer, such as what knowledge nodes need to make cooperative decisions.

The Contract-Net framework concentrates on a subset of CDPS goals and issues. In terms of goals (Table 1), the Contract-Net concentrates on allocating tasks to increase parallelism and to make effective use of network resources. It assumes that the allocated tasks are nearly independent, that is, managers will decompose tasks to minimize subproblem interactions; and it assumes that the manager will implicitly know how to integrate the results of its contractors. In short, the

Examples of task announcement, bid, and award messages are shown for the distributed sensor net application.

Figure 1: Contract Net: Messages

Contract-Net framework is geared toward top-down decomposition of large tasks and allocation of the subtasks. It is thus best suited for CDPS applications with well-defined task hierarchies, with tasks that are initially presented to a few nodes in the network (in contrast to tasks that are distributed over all the nodes in a network) and that can be decomposed into nearly independent subtasks. Partial global planning (discussed in Section 2.5.2) subsumes contracting in an approach that allows nodes to suitably decompose and distribute tasks and to cooperatively solve inherently distributed problems by communicating in a more expressive framework.

2.1.2 Multistage Negotiation

Another use of a limited form of negotiation in task allocation has been developed by Conry and her colleagues [12]. They consider a class of task allocation problems called **distributed constraint satisfaction** problems, in which a coordinated set of actions is required to achieve the goals of the network, but each node has only limited resources available for completing all of its assigned actions. The combination of local resource constraints and the need for coordination of actions among nodes gives rise to a complex set of global, interdependent constraints.

Conry and her colleagues investigated task allocation in the long-haul, transmission "backbones" of larger, more complex communications networks. These systems consist of networks of sites, each containing a variety of communications equipment, interconnected by links. Sites are partitioned into geographic subregions, with a single site in each subregion designated as a control facility. A control facility is responsible for monitoring and controlling its subregion. At any given time, the network may be supporting communication between several different sets of users. Each set of users is allocated a dedicated set of resources to create an end-to-end connection called a circuit. Normally, the circuit lasts for the entire duration of the communication. However, equipment failures or outages can break a circuit. The control facilities of each subregion are monitored for interruptions, and when they occur, the control facilities interact as a CDPS network to assess the situation and to cooperatively develop alternative dedicated circuits to restore end-to-end communication.

Multistage negotiation extends the basic Contract-Net protocol to allow iterative negotiation during the bidding and awarding of tasks. Nodes tentatively choose local actions to allocate and link communication resources, and iteratively exchange this information. At each iteration, each node assesses how its local choices and the current tentative choices of other nodes impact which circuits could be restored. Specifically, each node detects whether a choice it has made violates the expectations of another node concerning the use of resources to restore a circuit. Through the iterative exchange of relevant information, the nodes converge on compatible choices (or recognize that the problem is overconstrained). Through multistage negotiation, nodes exchange only enough information to find a configuration that satisfies their constraints, rather than insisting that each node have a global view of all nodes' choices and their resource utilization requirements.

2.1.3 Negotiation in Air-Traffic Control

Negotiation is an important part of the cooperation strategies developed by Cammarata, McArthur, and Steeb for resolving conflicts among plans of a group of nodes [7]. They have worked in the air-traffic control domain, where the goal is to develop CDPS techniques that will permit each node (aircraft) to construct a flight plan that will maintain an appropriate separation from other airplanes, and satisfy other constraints such as getting to the desired destination with minimum fuel consumption.

Their most well-developed approach to this problem is a policy they call **task centralization**. In this policy, airplanes involved in potential conflict situations (which occur when airplanes could become too close, based on their current headings) choose one of the agents involved in the conflict to resolve it. This airplane acts as a centralized planner to develop a multi-agent plan that specifies the concurrent actions of all of the airplanes (for more on multi-agent planning, see Section 2.4). Although this technique is not itself distributed, the airplanes do use negotiation to decide which of them should do the planning. The chosen airplane is sent the detailed plans of the other agents involved in the potential conflict and it attempts to modify only its own flight plan to resolve the conflict. The replanning airplane then transmits its revised flight plan to all agents that have received its earlier flight plan.

This replanning cycle iterates if one or more airplanes perceive a conflict based on the updated plan. This could occur if the replanning airplane could not completely resolve the conflict by modifying only its local plan or if the replanning airplane did not know that the new plan conflicts with the plans of airplanes not included in the original conflict set. The advantage of this centralized approach is that it avoids the possibility of agents generating inconsistent plans. In the decentralized case, agents can simultaneously change their plans (thus generating inconsistent views) so their plans might not avoid conflicts. The disadvantage of the centralized approach is that it may require many planning cycles to reach an acceptable solution. If too many iterations are needed, there may be insufficient time to carry out the plan. The centralized planning approach also requires that all the flight plans have to be communicated to a single replanning airplane, making that airplane a communication bottleneck and a reliability risk.

Using the task centralization policy, Cammarata and her colleagues evaluated a number of strategies for choosing a suitable airplane to do the replanning. These strategies considered factors such as a plane's room to manueveur, how much a plane knew about the flight plans of other planes, and how committed the plane was to its own flight plan. Their results indicated that choosing the most knowledgeable plane is usually a good strategy, except in very complex situations where that plane is overwhelmed with data, at which point the best choice is a plane with the most room to manueveur.

2.1.4 Cognitive Modeling of Negotiation

Although her research has not been directly applied to CDPS, Sycara is developing a computational model for multistage negotiation leading to compromise between multiple agents dealing with multiple issues in both single and repeated encounters [87, 84, 85]. Her model is based on messages that contain proposed compromises, persuasive arguments, reasons for agreement or disagreement with compromises and arguments, requests for additional information, and agents' utility measures for the issues they disagree on. She uses case-based reasoning techniques to drive the negotiation process, storing a history of past negotiations. When this approach fails, she relies on multi-attribute preference analysis to decide on the most likely compromise that will be acceptable to the other agents. Thus, as they develop a compromise, the agents dynamically revise their goals. Her techniques for forming compromises, and for maintaining a history of previous negotiations and compromises, could be useful in CDPS networks.

The negotiation framework developed by Sathi and his colleagues also has not been directly applied to CDPS. In their framework, agents iteratively relax their constraints until a compromise is reached [70]. Agents can propose a compromise using a variety of negotiation operators, such as slightly relaxing interacting constraints (log-rolling), agreeing to relax the solution criteria, substituting one resource with another, bridging the different viewpoints by proposing a completely new solution, overlooking weak interactions among constraints, or appealing to third parties to settle the disagreement. These operators would similarly be useful in a CDPS network for resolving conflicts and inconsistencies when problem-solving nodes have different but related subproblems to solve.

2.1.5 Summary

Negotiation is a complex and variable phenomenon, and to date CDPS researchers have only been able to study some of its specific forms. It is important to CDPS research because it is a natural way for systems to coordinate decisions to achieve several cooperative goals, including assigning tasks to increase parallelism and to effectively use network resources (Table 1). Recalling the more specific goals of cooperation (Section 1.2), negotiation allows nodes to improve the use of computing resources and expertise through exchanging tasks, to increase the solution creation rate through parallelism, and to increase the certainty of results or the likelihood of generating those results by assigning the same task to several nodes that may have different expertise.

2.2 Functionally-Accurate Cooperation

A recurrent problem in CDPS research is how to get nodes with inconsistent views and information to cooperate effectively. Inconsistencies arise because nodes might have incomplete or out-of-date views of the states of other nodes, contradictory raw information about related subproblems (e.g., different but error-prone reports of the same events), conflicting long-term problem-solving knowledge, different views of the network's goals, and errors in hardware or software. Yet, despite these inconsistencies, the nodes should cooperate if at all possible.

One approach to dealing with inconsistency is to not allow it in the first place, or at least to not consider it. For example, the Contract-Net protocol (Section 2.1.1) decomposed and distributed tasks in such a way that there was always a manager node to coordinate its contractors, who would in turn always pursue the task as expected. A second approach is to resolve inconsistency through explicit negotiation. For example, mechanisms for negotiation (Sections 2.1.2 and 2.1.3) allow nodes to recognize inconsistencies in the form of constraint violations, and to modify their plans accordingly.

A third approach is build CDPS networks that continue to perform despite inconsistency. A purely practical argument for this approach is that, in many applications, it is too expensive to implement the necessary communication, synchronization, and hardware reliability to guarantee that each node has a complete, consistent, and up-to-date view of the information it needs to solve its subproblems in a way that does not lead to inconsistency. A second argument is more theoretical: As CDPS networks become very large and complex, operate in the real world, and evolve over time, it is impossible to guarantee that knowledge among the nodes will remain consistent. In fact, it has been argued that some degree of inconsistency among nodes is beneficial to network problem solving [15, 63].

If the network permits inconsistencies, it must be able to resolve them. To do so, it should allow nodes to exchange inconsistent partial solutions, so that one or more of them has enough information to resolve the inconsistency. Exchanging partial solutions also helps prevent nodes forming inconsistent partial solutions, because a partial solution can represent **predictive information** that a recipient of this partial solution can use as context to predict the characteristics of partial solutions to build that will be consistent and compatible with it. The timely exchange of predictive partial solutions can thus reduce the search for compatible solutions and therefore make network problem solving faster (see also Section 2.5.1).

A node should also have the ability to tolerate inconsistency in its control information. In many applications, the information necessary to make informed local control decisions—decisions that are consistent with control decisions at other nodes—may not be located at the node making the decision. The costs to obtain consistent, complete, and up-to-date information might be prohibitively large in terms of delays due to synchronization and message transmission. Additionally, in a highly dynamic environment, the cost of recomputing network control for each minor change in the state of network may also be expensive [20]. Thus, it might be more cost-effective to tolerate some level of inconsistency in control information than to try to resolve the inconsistency, so long as the application domain allows a problem solver to recover from incorrect control decisions.

2.2.1 The Functionally-Accurate, Cooperative Approach

Lesser, Corkill, and Erman were among the first researchers to explore the possibility of building CDPS networks that worked effectively despite inconsistencies [51, 49]. In their **functionally-accurate, cooperative** (FA/C) approach, network problem solving is structured so that nodes cooperatively exchange and integrate partial, tentative, high-level results to construct a consistent and complete solution. A node's problem solving is structured so that it is not necessary for its local knowledge bases to be complete, consistent, and up-to-date in order to make progress in its problem-solving tasks. Nodes do the best they can with their current information, but their solutions to their local subproblems may be only partial, tentative, and incorrect.

Error resolution becomes an integral part of network problem solving as nodes try to combine (and assess the implications) of partial and tentative results received from other nodes. Thus, the advantages of the FA/C approach are accrued at the cost of making local problem solving more complex. If one is willing to pay this cost, then the FA/C approach can reduce the synchronization and communication among nodes that would be required to guarantee network consistency. Moreover, because FA/C networks can tolerate inconsistency, they can be more resilient and robust in face of processor, sensor and communication failures.

Lesser and Erman developed a three-node FA/C network of modified Hearsay-II speech understanding systems, in which each system sampled one time-continuous segment of the speech signal [51]. The systems exchanged only high-level intermediate results consisting of phrase hypotheses, and yet could still converge on complete interpretations despite the loss of some messages. The FA/C approach was subsequently studied in the context of a distributed interpretation task for vehicle monitoring (see Section 2.3.2), where the local data received by nodes from their sensors has limited scope and is potentially errorful. As this research progressed, it became clear that the unrestrained exchange of tentative partial results—a naive implementation of FA/C—quickly bogs down in the presence of large amounts of information. As described in Section 2.3, this led to techniques built on top of FA/C to control it.

2.2.2 Open Systems

Building on the FA/C approach and his previous work on the Scientific Community Metaphor [45], Hewitt has elaborated on the concept of Open Systems [39]. The Open Systems approach, which Hewitt has recently referred to as Organizational Knowledge Processing, emphasizes the need for agents to cope with conflicting, inconsistent and partial information as an integral part of their operation; and to be highly reliable so that the operation of the system is continuous. An open system consists of a network of microtheories in which an agent or a small set of agents can reason logically and maintain consistent knowledge within a microtheory. However, the network of microtheories taken as a whole can be inconsistent. Debate and negotiation are used to resolve these inconsistencies. Hewitt further assumes that the "internal operation, organization and state of one computational agent may be unknown and unavailable to another agent..." [39]. This assumption requires cooperative strategies that are similar to those needed to deal with heterogeneous expert systems, where individual systems may use different problem-solving strategies and knowledge representations and have different criteria for judging acceptability of solutions. The Open Systems approach is currently a concept with no implementation, but it seems to represent an important conceptual framework for structuring large and complex CDPS networks made out of heterogeneous agents that can both passively tolerate and actively address inconsistencies.

2.2.3 Summary

Functionally-accurate cooperation suggests that the exchange of tentative, partial results will allow nodes to eventually converge on correct and consistent larger results. More communication will generally reduce the inconsistency, because nodes will have more common information. Less communication will result in more inconsistency, which will in turn force the nodes to perform extra computation because they generate a larger number of inconsistent results and must spend computing resources on resolving inconsistencies. Thus, functionally-accurate cooperation highlights a basic tradeoff in CDPS networks between communication and computation. Nodes that are free to generate potentially incorrect and inconsistent tentative results can more completely explore the space of possible solutions. Functionally-accurate cooperation can, therefore, increase the set or scope of achievable tasks by sharing information (Table 1), and in doing so it achieves the more specific cooperative goal of increasing the variety of solutions by allowing nodes to form local solutions without being overly influenced by other nodes (Section 1.2).

2.3 Organizational Structuring

One important difference between negotiation (Section 2.1) and functionally-accurate cooperation (Section 2.2) is that negotiation takes a top-down view of problem solving while functionally-accurate cooperation takes a bottom-up view. Functionally-accurate cooperation allows nodes that are solving inherently distributed problems to work semi-autonomously, and through communication to build up overall solutions despite the fact that each is ignorant of how the others are contributing to overall solutions. The difficulty is that this ignorance can lead to excessive communication, duplication of effort among nodes, and generally ineffective use of network resources. In contrast, forms of negotiation such as contracting allow nodes to decompose and allocate their problem-solving responsibilities, to structure their actions and interactions to more effectively work as a team for a particular problem. The trouble with this approach is that, in applications where problems are inherently distributed, the nodes might not initially know how their local subproblems fit together into the larger network problems. That is, they might not know what particular problem they are all trying to solve. In addition, their perception of the problem can change in the course of problem solving, dictating a need to dynamically alter the cooperative relationships between nodes.

Organizational structuring attempts to find a compromise between the strongly top-down view of contracting and the bottom-up view of functionally-accurate cooperation. An **organizational structure** of a CDPS network is the pattern of information and control relationships that exist between the nodes, and the distribution of problem-solving capabilities among the nodes. Whereas a contracting approach dynamically defines the relationships between nodes to solve a specific problem, an organizational structure gives more general, long-term information about the relationships between nodes. That is, contracts represent temporary alliances (as in a construction project, where the relationship ends when the structure is built) while an organization is more permanent (as in a corporation, where the roles of the president and vice-president are stable for long periods of time). The organizational structure can augment a functionally-accurate, cooperative system to give each node a high-level view of how the network solves problems and the role that the node plays within this structure. With this general, high-level view, the nodes can ensure that they meet conditions that are essential to successful problem solving, including [15]:

Coverage. Any necessary portion of the overall problem must be within the problem-solving capabilities of at least one node.

Connectivity. Nodes must interact in a manner that permits the covered activities to be developed and integrated into an overall solution.

Capability. Coverage and connectivity must be achievable within the communication and computation resource limitations and reliability specifications of the network.

The organizational structure must specify roles and relationships to meet these conditions. For example, to ensure coverage the organizational structure could assign problem-solving roles to the nodes that make each node a specialist at a different type of subproblem. The organizational structure must then also indicate connectivity information to the nodes so that they can route subproblems to be solved to nodes that are able to solve them. On the other hand, the organizational structure might make the network more robust by assigning overlapping subsets of specialties to the nodes, so that no node is irreplaceable. The connectivity information should still allow nodes to redistribute subproblems, but must also allow nodes with overlapping specialties to avoid redundantly solving the same subproblem. Because the network might be able to solve problems in several different ways, it must have nodes that have the authority to decide on and enforce a particular approach. An organizational structure can specify authority and connectivity for the flow of information and control between nodes in terms of topologies, such as hierarchical, heterarchical, flat (lateral) structures, matrix organizations, groups or teams, and market or price systems (Figure 2). These examples illustrate the variety of information that an organizational structure can contain.

2.3.1 Ties to Organizational Theory

Theories about human organizations can provide insights to CDPS, and the work of Galbraith, March and Simon, Williamson, and Dewey is particularly relevant [15, 25, 28, 39, 45]. For example, Several possible organizations are shown, where for each the horizontal axis corresponds to locations and the vertical axis corresponds to information level (higher levels contain more completely processed information). In (a), the nodes are organized laterally (flat), so that each processes all of its data and they exchange results among themselves. In (b), the nodes are arranged in a hierarchy, where some nodes form partial results that are sent to middle managers that integrate them and pass their results on to higher-level managers. In (c), the nodes are organized in a matrix organization, where separate nodes are responsible for different processing levels, and their results are integrated by other nodes.

Figure 2: Control Topologies

Galbraith [26, 27] has developed a set of paradigms for redesigning an organizational structure to cope with the increased communication caused by uncertainty (such as unexpected events and errorful information). Galbraith draws upon March and Simon's work, which recognized the limited information processing capabilities of humans [55, 72, 73]. Called **bounded rationality**, this limitation applies to both the amount of environmental (sensory) information that can be effectively used to make decisions, and the amount of control that can be effectively exercised. Bounded rationality has serious implications on the quality of decisionmaking under uncertainty, for "the greater the task uncertainty, the greater the amount of information that must be processed...to achieve a given level of performance" [27]. A motivation for variations in organizational structures (in terms of the type, frequency, and connectivity pattern on information flow) is to provide additional information processing capacity (to handle greater uncertainty) within the bounded rationality of the organization's individual members.

In CDPS, problem solving nodes also have limited information processing capabilities. Fox concentrates on the effects of complexity and uncertainty on choosing suitable CDPS organizations [24, 25]. For example, he says, "Complexity and uncertainty are two opposing forces; complexity forcing a distribution of tasks ultimately resulting in heterarchical structure; uncertainty pushing in the opposite direction, vertically integrating tasks into a more hierarchical structure." Malone and Smith have also tried to develop theories about how distributed networks should be organized [53]. They have used queuing theory models to analyze generic organizational classes to determine their performance strengths and weaknesses with respect to processing, communication, and reliability. Their analysis has shown that different organizational classes are appropriate given different problem situations and performance requirements, and that these characterizations change with the size of the organization. However, the implications of their work for CDPS are unclear, because they assume that the individual tasks are independent and because they do not model uncertainty in terms of the accuracy of control and problem-solving decisions.

Gasser [28] is pursuing a view of organization for CDPS that is less structural in perspective and more related to current organization theory [52, 83]. He views an organization as a "particular set of settled and unsettled problems about belief and action through which agents view other agents. Organizational change means opening and/or settling some different set of questions in a different way, giving individual agents new problems to solve and ... a different base of assumptions about the beliefs and actions of other agents" [30]. Gasser asserts that agents need the ability to recognize, diagnose and repair violated expectations when other agents fail to meet default assumptions in previously settled questions. When expectations are violated, agents generate new questions that must be settled. Agents also require the ability to perform meta-level reasoning, change their goals and settle the associated set of questions that now need to be solved in order to effectively carry out their new goals. This latter capability leads to organizational change while the former is more associated with local refinements of the organization. Gasser is currently using the MACE testbed [29] to simulate a game-like application domain where agents of one type attempt to surround and capture agents of another type [4]. He is using this simulated application to empirically evaluate his coordination mechanisms [30].

2.3.2 Coordination Using Organizational Structuring

Corkill and Lesser have applied organizational structures to CDPS to efficiently implement network coordination strategies. They use an organizational structure to limit the range of control decisions made by nodes (decreasing the demands on a node's information processing capabilities) and to ensure that information necessary for making informed decisions is routed to the appropriate nodes. The organizational structure provides a control framework that increases the likelihood that the nodes will work as a coherent team by providing a general and global strategy for network problem solving.

They have implemented and evaluated their ideas in one of the most flexible CDPS simulation testbeds developed to date: the Distributed Vehicle Monitoring Testbed (DVMT) [15, 50]. The DVMT simulates a network of nodes that perform distributed interpretation to track vehicles moving among them. The spatially-distributed nodes detect the sounds of vehicles, and each applies knowledge of vehicle sounds and movements to track vehicles through its spatial area. Nodes then exchange information about vehicles they have tracked to build a map of vehicle movements through the entire area. This task requires functionally-accurate cooperation because nodes work with potentially errorful data, leading them to generate potentially inconsistent and incorrect partial results. The nodes can converge on acceptable overall solutions if they exchange enough of their partial results. However, without an organizational structure to guide their processing and communication decisions, the nodes could quickly overwhelm each other with tentative partial results.

Corkill and Lesser recognized that, in designing a CDPS network to exploit organizational structures, they must have nodes with substantial sophistication. They suggest that it is unrealistic to expect to develop network-wide control policies that are sufficiently flexible and efficient, and require limited communication, while simultaneously making all the control decisions for each node in the network. Instead, each node needs to decide on its own activities based on its current local view of the problems being solved, but to use organizational knowledge about its problem-solving role in the network and the roles of other nodes to guide its decisions, so that it is a more effective participant in the network. This approach divides the problem of network coordination into two concurrent activities [14]: the construction and maintenance of a network-wide organizational structure; and the continuous local elaboration of this structure into precise activities using the local knowledge and control capabilities of each node. Thus, within the general bounds specified by the organizational structure, the nodes have substantial latitude as to what decisions they make.

In the DVMT, an organization is specified as a set of "interest areas" associated with each node that define what, when, and to whom information (partial results and goals to build partial results) should be transmitted, authority relationships indicating how much priority nodes should give to processing externally received goals versus internally generated goals, and goal priorities that indicate how to evaluate the importance of processing different types of goals.

Each node in the DVMT is a blackboard-based problem solver, with levels of abstraction and knowledge sources appropriate for vehicle monitoring (Figure 3). A DVMT node's scheduler, which decides what knowledge source will be applied to the partial results on the blackboard, has been modified to use interest area specifications to prioritize the goals for generating different partial results. Goals (and subgoals of those goals) to generate results in areas of high interest for the node have their priority raised. For example, to increase the range of tasks a node could perform, the organization might allow it to interpret data from two different sensed areas, A and B. A neighboring node might also receive data for sensed area B. Thus, to avoid duplication of effort, the interest areas would specify that the node should prefer to achieve goals that generate partial solutions in area A, but if it has no such goals then it can pursue goals in area B to cooperate with its neighbor to process the data in parallel. Similarly, the ratings of goals to transmit and receive information are influenced by the interest areas of both the sending and receiving nodes.

Without interest areas, a node would simply pursue its highest-rated goal. This rating would be based on factors such as the confidence in the partial results that triggered the goal. The interest areas are themselves rated, and so when a node incorporates interest area information, it re-rates a goal based on its initial rating combined with its interest area's rating. Thus, the organizational structure only biases the node's decisions: because goal ratings also involve other factors, a node could pursue a goal that is highly-rated from a local perspective even if it falls in a poorly-rated interest area. The organizational structure thus provides guidance without dictating local decisions, and can be used to control the amount of overlap and problem solving redundancy among agents, the problem-solving roles of the nodes (such as "integrator," "specialist," and "middle manager"), the authority relations between nodes, and the potential problem-solving paths in the network.

Durfee, Lesser, and Corkill expanded these ideas and showed that a static organization structure cannot always guarantee coherent network behavior [21]. An organization that is specialized for one short-term situation may be inappropriate for another. Because network reorganization is assumed to be costly and time consuming, and since specific problem characteristics cannot be predicted beforehand, an organizational structure should be chosen that achieves acceptable and consistent performance in the long-term, rather than very good performance in only a few situations. An acceptable organizational structure will provide nodes with enough flexibility for them to react suitably to changing situations. The trouble with flexible organizations is that they also give nodes the latitude to make decisions that might lead to incoherent network behavior. Given the range of roles they *could* play in an organization, nodes need to solve the problem of what roles they *should* play in the current situation. In essence, this is another CDPS problem that nodes should solve together, if they are to work effectively as a team. Durfee and Lesser expanded on the organizational structuring approach to develop a meta-level organization for specifying how nodes in a CDPS network should cooperatively decide how to solve a problem together [19]. That is, the meta-level organization organizes the coordination activities of the nodes, while the separate domain-level organization organizes their domain-level problem-solving activities.

An unanswered question for this research is where organizational structure comes from in the first place. To date, they have all been developed by the human user and initialized with the network. A long-term goal of the research has been to automate the organizing process, so a network can reorganize when necessary. Reorganization is a costly process, both in communication and computation, and should only be undertaken infrequently, and only when problems arise with the current organization. Detecting problems with the current organization is a difficult task, because performance measures are distributed among the nodes: perhaps no node has enough information to recognize when the organization is no longer effective. The work by Durfee and Lesser on meta-level organizations and on treating coordination as CDPS might provide tools for addressing the recognition problem. That is, if the coordination problem becomes overly difficult, it might indicate that the underlying domain-level organization is at fault. Once the nodes determine that they should reorganize, negotiation techniques could provide the basis for converging and agreeing on a new organization. Of course, the problem of detecting poor meta-level organizations remains.

The DVMT problem-solving architecture is depicted, indicating how organizational roles are considered when goals are processed, affecting the ratings of KSIs (possible knowledge source executions). The planner decides which KSI to invoke, triggering the knowledge source, which generates new hypotheses on the data blackboard. Finally, the goal processor builds goals to improve on these hypotheses, and uses the organizational structure to influence the ratings of these goals.

Figure 3: Organizational Structuring Architecture

2.3.3 Other CDPS Organizations

The contracting approach of Davis and Smith (Section 2.1.1) allows nodes organize themselves to solve particular problems. When the network receives a large problem, the nodes recursively decompose the problem until nondecomposable tasks remain, assign subproblems to nodes through the Contract-Net protocol, solve the subproblems in parallel, and synthesize one or more answers from the subproblem solutions. Synthesizing the answer might require the nodes to locate and collect the best solution to each subproblem, because several nodes might solve the same subproblem in different ways. Contracting allows nodes to form simple organizations in terms of manager-contractor relationships. As nodes solve one problem and move onto another, they reorganize themselves. Thus, contracting provides an example of automated network organization, but the overhead of having to reorganize for each problem could be substantial.

From a different perspective, Kornfeld and Hewitt have proposed that CDPS can be organized in a manner analogous to the structure of scientific research [45]. In their **scientific community metaphor** for CDPS, nodes posit either "questions" (goals) or "answers" (results) into a mutually accessible archive. The presence of this information allows a node to draw on work already performed by other nodes. Kornfeld and Hewitt implemented a parallel processing version of this system, called Ether, that embodies many of their ideas [44]. However, although the metaphor is an interesting way of viewing CDPS networks, significant research on aspects of the implementation in a distributed environment remain.

2.3.4 Summary

Imposing a general, high-level organization on a CDPS network gives nodes knowledge that improves how they coordinate, while still allowing them to pursue alternative solution paths that are not dictated by the network. The organization helps nodes focus their processing and communication resources on activities that are usually more likely to lead to effective network performance, although in any given problem-solving situation a more rigid, situation-specific organization could lead to even better performance. If the characteristics of the problem-solving situation are known beforehand, then techniques such as contracting, which allow nodes to generate a situation-specific organization, are suitable. However, because CDPS networks often work in dynamic domains where the problem-solving situation can frequently change, organizational approaches have also focussed on general-purpose organizations.

Organizational structures help nodes to cooperate, and thus help to achieve the goals of cooperation (Table 1). Organizations allow nodes to work in parallel. Appropriate organizations assign network problem-solving roles to nodes with suitable resources, so that network resources are used effectively. If specified in the organization, nodes can have overlapping roles to increase network reliability. Finally, the organization gives each node a global picture of the problem-solving roles of the other nodes, so that each node can determine for itself whether any of its local activities might interact with those of another node. But note that some organizations, such as those based on simple contracting relationships might not notice potential interactions between nodes, because a manager only knows about its contractors, and not about all of their subcontractors.

2.4 Multi-Agent Planning

In a multi-agent planning approach to cooperation, nodes (agents) form a multi-agent plan that specifies all of their future actions and interactions. Coordinating nodes through multi-agent plans is different from other approaches in that one or more nodes possess a plan that indicates exactly what actions and interactions each node will take for the duration of network activity. This differs from approaches such as contracting, in which nodes typically make pairwise agreements about how they will coordinate, and nowhere is any complete view of network coordination represented. Contracting can lead to incoherent behavior that multi-agent planning avoids. For example, in contracting, two different nodes might independently form and contract out the same subproblem to two other nodes, so that these nodes are duplicating each other's effort because they are unaware of contracts in which they do not participate. In multi-agent planning, one or more nodes would have information about each node's activities and could recognize and prevent the duplication of effort. Because it insists on detecting and avoiding inconsistencies before they can occur, multi-agent planning is not like functionally-accurate cooperation. Finally, unlike the general guidelines imposed by an organizational structure, a multi-agent plan dictates exactly what actions each node should take and when.

A multi-agent plan is built to avoid inconsistent or conflicting actions, and is typically used in CDPS networks to identify and plan around resource conflicts. For example, in scheduling the use of airspace [7] or machining tools [31], an unexpected conflict for a resource can be costly in time, money, or human life. Rather than risking incoherent and inconsistent decisions that nodes might make using other approaches, multi-agent planning insists that nodes plan out beforehand exactly how each will act and interact. Because multi-agent planning requires that nodes share and process substantial amounts of information, it can also require more computation and communication resources than other approaches.

2.4.1 Centralized Multi-Agent Planning

Georgeff develops a multi-agent planning approach where the plans of individual nodes are first formed, and then some central planning node collects them, and analyzes them to identify potential interactions, such conflicts between the nodes over limited resources [31]. The central node then performs a safety analysis to determine which potential interactions could lead to conflicts; for example, when a node modifies the world in such a way that another cannot continue with its plan. The central planning node next groups together sequences of unsafe situations to create critical regions. Finally, it inserts communication commands into the plans so that nodes synchronize appropriately. For example, if one node should wait until another has finished with a lathe before it begins operating the lathe, then its plan contains an instruction to wait for a message from the other node before beginning to use the lathe. Georgeff and Lansky have pursued this centralized multi-agent planning approach further, using alternative representations for events in multi-agent domains [32, 33, 47].

Cammarata, Steeb, and McArthur have also developed a system for centralized multi-agent planning [7]. Their system, described more completely in Section 2.1.3, works in an air-traffic control application. Through a process of negotiation, the nodes (aircraft) choose a coordinator. Each node then sends this coordinator relevant information, and the coordinator builds a multi-agent plan that specifies all of the nodes' planned actions, including the actions that it, or some other node, should take to avoid collisions.

2.4.2 Distributed Multi-Agent Planning

When multi-agent planning is done in a distributed manner, there may be no single node with a global view of network activities, so detecting and resolving interactions between nodes is much more difficult. The general approach is to provide each node with a model of other nodes' plans [13, 32, 43]. For example, Corkill has developed a distributed hierarchical planner based on NOAH [69] where nodes represent each other using MODEL nodes [13] and synchronize the plan execution with explicit synchronization actions. The nodes plan together level by level. They each build local plans at one level of detail, and build suitable models of each other by communicating about shared resources needed for their goals. The nodes resolve resource conflicts using a protocol that Corkill has developed for a distributed version of NOAH's critcs. When conflicts in their plans at this level have been resolved, the nodes then proceed to the next level of detail and repeat this process.

Using a logic-based approach, Rosenschein and Genesereth studied how agents with a common goal but different local information can exchange propositions to converge on identical plans. In their formulation, goals are propositions to prove, local information is represented in axioms, and plans are proofs of the goal propositions[67]. They developed strategies for convergence based on assumptions about the correctness and completeness of agents' information, about what each agent knows about other agents' knowledge, about what each agent knows about itself, and about whether additional information can cause a previously acceptable plan to be unacceptable. In cases where agents might have incorrect information, or incorrect views of the information of other nodes, Rosenschein and Genesereth show that convergence on a plan cannot be guaranteed, whatever strategy is used. Their results indicate that expecting sometimes unpredictable agents working in dynamic domains to always coordinate optimally is infeasible; perhaps the best we can hope for is that they will coordinate acceptably well and will tolerate any uncoordinated activity.

Durfee and Lesser have developed a CDPS approach called partial global planning (Section 2.5.2), in which nodes build local plans and share these plans to identify potential improvements to coordination [19]. Unlike multi-agent planning, which assumes that a plan is formed before nodes begin to act, partial global planning allows nodes to interleave planning and action. In partial global planning, nodes coordinate as best they can given their current view, rather than waiting for a complete view of the network. This ability is essential in dynamic domains where complete, up-to-date information might not be available to any node. Unlike conventional multi-agent planning, partial global planning also gives nodes flexibility in deciding what node or nodes will build larger plans (represented in the meta-level organization discussed in Section 2.3.2).

2.4.3 Summary

Multi-agent planning takes the view that interactions between the separate activities of the nodes must be identified, and any conflicts should be identified and fixed before the plans are executed. This view is critical in applications such as air-traffic control, where aircraft should wait until all of their situations are taken into account in a plan before they begin executing the plan by changing course. Thus, typical multi-agent planning approaches concentrate mostly on the particular cooperative goal of avoiding harmful interactions (Table 1). On the other hand, multi-agent planning systems are poorly suited to dynamically changing domains, where nodes cannot wait for complete information about potential plan interactions before they begin acting. In most dynamic domains, nodes can make mistakes—take actions that are not well coordinated—without catastrophic results, and can later take actions to rectify any problems resulting from earlier actions.

For these domains, partial global planning can be a better approach (Section 2.5.2).

2.5 Sophisticated Local Control

A node in a CDPS network must be more sophisticated than a node that works alone, because it must reason about its own problem solving, how this fits in with problem solving by other nodes in the network, and what it can do to improve network problem solving. It is a mistake to assume that effective coordination will result if we take nodes that are individually good problem solvers and then simply give these nodes a communication interface so that they can exchange messages. Coordination is not achieved just through exchanging information; nodes must reason about what that information represents and how exchanging information will affect their individual and group behavior.

Sophisticated local control allows a node to understand the implications of its planned problem-solving and communication actions on other nodes' goals, beliefs, and plans. This understanding forms the basis for deciding how to coordinate with others, such as whether to contract out tasks, negotiate over how to achieve goals, exchange information to resolve inconsistent views, take on different organizational responsibilities, or plan specific actions and interactions to build complete solutions. Sophisticated local control concentrates on how to build nodes that can decide for themselves how and when to coordinate, rather than having a specific coordination approach imposed on them.

2.5.1 Communication Policies

Because nodes are influenced by the information they receive, they need policies to guide their decisions about what information to exchange, with whom, and when. Durfee, Lesser, and Corkill describe three major characteristics of the information communicated among nodes that affects global **coherence**—how well the nodes work as a team. These are relevance, timeliness and completeness [21]. The **relevance** of a message measures the amount of information that is consistent with the solution derived by the network. Irrelevant messages may redirect the receiving node into wasting its processing resources on attempts to integrate inconsistent information, so higher relevance of communicated information can result in more global coherence because it stimulates work along the solution path. The **timeliness** of a transmitted message measures how much it will influence the current activity of the receiving node. Since timeliness depends not only on the content of the message but also on the state of the nodes, a message's timeliness can vary as node activity progresses. If the transmitted information will have no effect on the node's current activity, there is no point in sending it; however, if the transmitted information will distract the receiving node to work in a more promising area, or if the node needs the information to continue developing a promising partial solution, then it is important that the information be sent promptly. Finally, the **completeness** of a message measures the fraction of a complete solution that the message represents. Completeness affects coherence by reducing the number of partially or fully redundant messages communicated between nodes—messages which negatively distract nodes into performing redundant activity. Furthermore, as the completeness of received messages increases, the number of ways that the messages can be combined with local partial results decreases due to their larger context. Achieving completeness is important to minimize communication requirements in our loosely-coupled distributed network.

These characteristics of communicated information are not independent. For example, higher completeness leads to higher relevance but, potentially, to a decrease in timeliness. Communication policies that guide decisions about what information should be sent, to what nodes, and when, often involve tradeoffs among the three characteristics. With increased self-awareness, a node can be more informed about the relevance and completeness of its local hypotheses and can make more intelligent predictions both about how a hypothesis will affect its local decisions and about whether the timely transmission of the hypothesis is therefore likely to cause other nodes to alter their activities.

Durfee, Lesser, and Corkill evaluated several communication policies in the DVMT (Section 2.3.2). When deciding whether to send a partial solution to a relevant node (as indicated by the organizational structure), a node could consider whether the partial solution is the first to be developed in processing new data (and so might provide predictive information to another node) or whether the partial solution is the last that can be locally generated for some data (and so represents a locally complete result to be integrated with results from other nodes). In evaluating different communication policies, they found that a policy's effectiveness depends on characteristics of the problem situation, such as how much the nodes' subproblems overlap and how much data each node has. Overall, the experiments showed that making nodes aware of which of their partial solutions are locally complete, and giving them knowledge about the potential impact of sharing their preliminary partial solutions, allows the nodes to communicate less and still coordinate better.

2.5.2 Partial Global Planning

Durfee and Lesser have emphasized the need for sophisticated local control to make reasoning about coordination an integral part of a node's local decisionmaking. Instead of having separate mechanisms for different forms of coordination, they have developed a unified, flexible framework in which nodes can form contracts, plan their actions and interactions, negotiate over their plans, use organizational information to guide their planning and problem-solving decisions, tolerate inconsistent views, and converge on acceptable network performance in dynamically changing situations despite incomplete, inconsistent, and out-of-date information. Their partial global planning approach [19, 18] thus addresses the different goals of cooperation (Table 1).

In the partial global planning approach, 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 maintains its own set of PGPs that it may use independently and asynchronously to coordinate its activities. Each PGP is a general frame-like structure for representing coordinated activity, using slots for representing nodes' goals, actions, interactions, and relationships.

Besides their common PGP representation, nodes also need at least some common knowledge about network problem-solving responsibilities and about how and when they should use PGPs to coordinate their activities. This common knowledge is represented in the domain-level and meta-level organizations (Section 2.3.2). Nodes use the domain-level organization to influence what goals they pursue and their plans to pursue them, and use the meta-level organization to decide how, when, and where to form and exchange PGPs based on their local plans. Guided by the meta-level organization, nodes use transmitted PGPs to build models of each other. A node uses its models of itself and others to identify when nodes have PGPs whose objectives could be part of some larger network objective, called a **partial global goal**, and combines the related PGPs into a single, larger PGP to achieve it.

Given the more complete view of group activity represented in the larger PGP, the node can revise the PGP (and afterwards, its local plans) to represent a more coordinated set of group actions and interactions and a more efficient use of network resources. For example, a PGP could indicate that a certain partial solution to be formed by one node could provide useful predictive information to another node. This expectation, and the transmission of the partial solution, are explicitly represented in the PGP, and indicates a plan to use information resources more effectively. As a second example, nodes that are working on the same network goal might have different PGPs, reflecting their different local perspectives. These nodes could exchange PGP information so as to negotiate over a compromised, agreed upon PGP. As a third example, a node could survey its current view of network PGPs and identify nodes whose computing resources or expertise are being underutilized. At the same time, other nodes could be overwhelmed with subproblems. By modifying its PGPs, the node could propose how the nodes could transfer appropriate subproblems to work as a better team. It sends these PGPs to potential subproblem recipients, who in turn can accept or reject the PGP, or modify it and send it back as a counterproposal. Thus, unlike a simple contracting protocol, partial global planning allows nodes to barter using proposals and counterproposals, where each proposal contains information about not only the subproblem to be transferred but also how that subproblem fits into network problem solving. This information helps a potential recipient make more informed decisions about how to respond to the proposal.

Partial global planning provides a unified framework that supports different forms of coordination, and has been implemented and evaluated in the DVMT (Section 2.3.2) [18]. In this implementation, a node's local planner develops a plan at multiple levels of detail, including a representation of major plan steps. In the DVMT, a major plan step corresponds to extending a partial track into a new time frame (such as extending the track formed from d_i–d_j into d_{j+1}, where d_k is data sensed at time k). This step might take several processing actions to analyze the new data, filter out noise, and integrate the correct data into the track. For each major plan step, the local planner roughly estimates what partial results will be formed and when. By representing and coordinating their major plan steps, nodes cooperate effectively without reasoning about details that are frequently revised and quickly outdated.

Each node has a partial global planner (PGPlanner) as an integral part of its control activities. The PGPlanner builds a node-plan from each local plan, where a node-plan's objective indicates the possible track(s) being developed and its plan-activity-map is a sequence of plan-activities. Each **plan-activity** represents a major plan step, and has an expected begin time, end time, and partial result, derived from the local planner's estimates. Guided by the meta-level organization, nodes exchange PGPs and node-plans so that one or more of them develops more encompassing PGPs. When combining PGPs into a single, larger PGP, a node merges the smaller PGP's plan-activity-maps to represent the concurrent activities of all participating nodes, and can reorder the plan-activities to improve coordination. It also builds a solution-construction-graph to indicate which partial tracks formed by the plan-activities should be exchanged to share useful information and construct the complete solution. The PGPlanner then revises local plans based on the PGP, and can propose transfers of subproblems to initiate negotiation that will lead to better use of network resources.

An important idea exemplified by the partial global planning approach is the distinction between "satisficing" network control and optimal network control. In environments which are highly dynamic and uncertain, and where an updated and consistent global view of the state of the network problem solving is very difficult to obtain, attempting optimal control at every moment is infeasible from both a computational and communicative perspective. Rather, partial global planning employs heuristic algorithms for reordering and revising PGP activities that achieve a reasonable balance between the interdependent requirements of global coherence, limited use of computational resources in controlling coordination, and responsiveness to dynamically changing conditions.

Two other ideas in the partial global planning framework contribute to our understanding of network coordination. The first is that increasing a node's understanding of its own activities is an important ingredient in designing effective coordination strategies. Durfee and Lesser show that providing a local node with the ability to develop high-level problem-solving goals and plans, make reasonably accurate predictions of the time required to achieve its planned steps, and make predictions about likely future goals, all lead to more sophisticated network coordination. The second contribution is the concept of network coordination as a distributed problem-solving task in its own right, distinct from domain-level CDPS going on among nodes. Partial global planning introduces the concept of a meta-level organization to describe the organizational relationship among nodes required to solve the network coordination problem and permits the coordination tasks to go on asynchronously and in parallel with domain problem solving.

2.5.3 Summary

A node that by itself is a good problem solver will not necessarily be a valuable participant in a CDPS network. Coordination requires that a node have more sophisticated local control so that it can more fully reason about goals and plans, both its own and those of other nodes. This view allows a node to cooperate and communicate more effectively, allowing it to influence other nodes and be influenced by them so that they work as an effective team. Sophisticated local control thus opens nodes up to a wide range of capabilities, including being able to negotiate, form contracts, develop plans, conform (or rebel against) organizations, and share results. In essence, sophisticated local control is the foundation on which more complex forms of coordination activity must be built, and allows the network to address all of the goals of cooperation (Table 1, as exemplified in partial global planning.

2.6 Formal Frameworks

Although an orientation toward techniques for particular application domains has dominated CDPS research, a number of researchers have instead concentrated on formal models of CDPS, using logic-based or game-theoretical nodes. Some of this work focuses on how nodes can form multi-agent plans, including the work of Georgeff and of Rosenschein and Genesereth (see Section 2.4).

CDPS requires that the formalisms developed for logic-based agents that work alone must be extended in two ways. The first extension is that these systems must be able to model and reason about the concurrent activities of multiple agents, as discussed in Section 2.4. The second extension is that the agents must perform in situations where they have incomplete knowledge or limited computational resources. Both cases lead to the possibility of generating incorrect inferences which in turn may result in agents having inconsistent beliefs about the world. As a result, agents might never converge on shared, coordinated plans [67]. Hewitt, in his studies of open systems (Section 2.2.2), addresses this problem and argues that formal logic is inadequate [39].

Researchers are following a number of different approaches to extending logical formalisms for CDPS applications. Konolige has developed the Deductive Belief model in which an agent's beliefs are described as a set of sentences in formal language together with a deductive process for deriving the consequences of those beliefs [42, 43]. This approach can account for the effect of resource limitations on the derivation of the consequences of beliefs. Appelt has used a possible world formalism to represent and reason about belief [2]. Cohen and Levesque have developed a formal theory for reasoning about an agent's intentions as combination of what it has chosen and how it is committed to its choice [10]. Rosenschein has developed a more general theory of multi-agent planning which allows for the existence of other agents and their mental states as part of the environment within which plans can be constructed [68]. Halpern and Moses have investigated the issue of common knowledge between agents, discovering limitations in what agents can know about each other [36].

Research on dialogue comprehension in natural language understanding is also relevant to CDPS research because both research areas must reason about multiple agents with distinct and possibly contradictory mental states [1, 9]. Mental states include not only facts or knowledge but also beliefs and goals. An agent must interpret messages from other agents, including what the messages imply about the agents' mental states, and must generate messages to alter the mental states of other agents, taking into account the potential actions of other agents that might affect how it can achieve its goals. Through an appropriate dialogue, the agents can converge on shared plans for how they should coordinate their activities [35, **?**].

Another research approach towards developing a formal theory for understanding the nature of cooperation among multiple agents is that of Rosenschein and Genesereth [65, 68, 66]. They have based their model on game theory techniques and have shown the utility of communication to resolve conflicts among agents having disparate goals. Using a game-theoretic model, each agent attempts to choose an option to maximize its payoff, and since no combination of agents' options might lead to maximal payoffs for them all, they must somehow arrive choose options that lead to acceptable payoffs given the circumstances. Rosenschein and Genesereth study how different assumptions about the rationality of the agents can lead to more or less effective choices. Certain assumptions about rationality allow agents to make reasonable choices without communication (since they each have complete information about the choices and payoffs for every agent), while the ability to communicate allows them to make deals about mutually beneficial activity in situations where complete information about the payoff matrix is not enough.

In summary, formal CDPS approaches attempt to use rigorous models of agent reasoning and interactions to develop insights into coordination that are independent of any domain. Many of these insights are useful to researchers attempting to build CDPS systems, illuminating crucial issues and limitations in what can be expected from CDPS networks. To remain tractable, however, the formal approaches often use simplified views of agents and their knowledge, such as assuming that all of the choices and payoffs of agents are known in advance. Because of the complexity gap between the systems being modeled formally and the applications that are being studied, CDPS research has yet to adequately define rigorous approaches that work in real-world applications.

3 Conclusion

The promises and pitfalls of CDPS can be summed up by combining two proverbs: Many hands make light work, but too many cooks spoil the broth. CDPS networks where nodes work together effectively have many potential benefits in applications where information, resources, or expertise are naturally distributed, or where they can be intentionally distributed to improve the speed, modularity, or reliability of the system. These benefits will not be realized, however, if our CDPS networks are uncoordinated.

We have outlined many approaches for coordinating nodes in a CDPS network, including contracting, negotiation, organizational structuring, multi-agent planning, and sophisticated local control. From these very different approaches, we infer that effective coordination requires three things. First, it requires structure, because without structure the CDPS nodes cannot interact in predictable ways. Structure is embodied in shared information such as organizations and communication protocols. Second, effective coordination requires flexibility, because CDPS nodes typically exist in dynamically changing environments where each node might have incomplete, inaccurate, or obsolete information. Flexibility allows a contracting node to decide how to bid in its current situation, it allows a node in an organization to locally decide what partial solution to form given its current data, and it allows a node in a planning system to change its plan in response to changing circumstances.

The third requirement for effective coordination is the knowledge and reasoning capabilities to intelligently use the structure and flexibility. Nodes must form and reason about what they are doing—their goals, plans, and beliefs—and how this fits into what they know about others. They must rely on structure to guide this reasoning, but must allow themselves the flexibility to adapt their activities to changing circumstances. In short, nodes need enough local sophistication to steer an appropriate course between regimentation and anarchy.

None of the approaches detailed earlier represents a general answer to the needs of intelligent coordination in CDPS networks, but our discussion has illustrated the richness of the ideas and approaches to date. Future CDPS research will build on this past work in many directions. One will be to improve our theories about organizing CDPS networks. These theories should provide guidelines for how domain and control problem-solving tasks should be distributed among agents based on the current network characteristics and problem-solving situation.

CDPS research will also extend its paradigms for how to get disparate agents to cooperate on a problem. These paradigms must address issues of how to resolve inconsistencies due to agents' different problem-solving approaches or knowledge, how to promote understanding between disparate agents of their beliefs, goals, and plans, and how to get agents to make intelligent communication decisions that influence each other to their advantage.

To incorporate CDPS into AI practice, we will need guidelines or frameworks for building AI systems that can become part of a CDPS network. If reasoning about coordination is an integral part of an agent, the agent must have knowledge representations, inference techniques, and control components that are adequate for this type of reasoning. Practical CDPS networks will also require advanced software infrastructures, languages and operating systems [5, 37].

CDPS has opened a new door in the study of intelligence. CDPS research continues to reveal the complexity of coordination in all of its forms, and the extensive though sometimes subtle connections between intelligent coordination and other aspects of intelligence. CDPS research brings to the fore issues in areas such as introspection, planning, language, and reasoning about belief. As Nilsson predicted in his early involvement in CDPS, research into CDPS forces us to address many of the basic problems of AI [59].

These insights have led some CDPS researchers to view reasoning about coordination—about how to interact with other intelligent agents—as a fundamental aspect of intelligent behavior. In fact, it could be argued that we judge the intelligence of an entity by how it interacts with us: whether we can understand its goals, plans, and beliefs as embodied in its actions, whether we can communicate with it, and whether it appears to be understanding us. CDPS research continues to study the knowledge and reasoning capabilities that must go into AI systems, if those systems are ever to meet these criteria for intelligence.

Further Reading

For further information on CDPS, the collection *Readings in Distributed Artificial Intelligence* edited by Bond and Gasser contains the seminal papers in the field [6]. In addition, the book *Distributed Artificial Intelligence*, edited by Huhns, provides papers on current research directions [40]. A second such book is due out late in 1989.

Acknowledgements

We would like to thank Clive Dym for helping us in our early drafts and Paul Cohen for many editorial suggestions.

References

[1] James F. Allen. *A Plan-Based Approach to Speech Act Recognition.* PhD thesis, University of Toronto, February 1979. (Also published as Technical Report 131/79, Department of Computer Science, University of Toronto, Toronto, Canada, February 1979.).

[2] D. E. Appelt. Planning natural language utterances to satisfy multiple goals. Technical Note 259, SRI International, Menlo Park, California, 1982.

[3] Ronald C. Arkin, Edward M. Riseman, and Allen R. Hanson. ArRA: An architecture for vision-based robot navigation. In *Proceedings of the DARPA Image Understanding Workshop*, pages 17–431, Los Angeles, California, February 1987.

[4] M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA, August 1985.

[5] R. Bisiani. A software and hardware environment for developing AI applications on parallel processors. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 742–747, Philadelphia, Pennsylvania, August 1986. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 451–456, Morgan Kaufmann, 1988.).

[6] Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[7] 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.).

[8] Ernest Chang. Participant systems. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 11, pages 311–339. Pitman, 1987.

[9] Philip R. Cohen. *On Knowing What to Say: Planning Speech Acts.* PhD thesis, University of Toronto, January 1978. (Also published as Technical Report 118, Department of Computer Science, University of Toronto, Toronto, Canada, January 1978.).

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

[11] S. Conry, R. Meyer, and J. Searlemen. A shared knowledge base for independent problem solving agents. In *Proceedings of the IEEE Expert Systems in Government Symposium*, Mc Lean, Virginia, October 1985.

[12] 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 Kaufman, 1988.

[13] 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. (An extended version was published as Technical Report 79-13, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, February 1979.).

[14] 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 appeared in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986).

[15] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* PhD thesis, University of Massachusetts, February 1983. (Also published as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.).

[16] W.B. Croft and L.S. Lefkowitz. Knowledge-based support of cooperative activities. In *Proceedings of the Twenty-first Annual Hawaii International Conference on System Sciences*, volume 3, pages 312–318, 1988. (Published by IEEE Computer Society Press, Catalog Number 88TH0213-9.).

[17] Keith S. Decker, Edmund H. Durfee, and Victor R. Lesser. Evaluating research in cooperative distributed problem solving. In Michael N. Huhns and Les Gasser, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*. Pitman, 1989.

[18] Edmund H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, 1988.

[19] 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*, Alan H. Bond and Les Gasser, editors, pages 285–293, Morgan Kaufmann, 1988.).

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

[21] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, November 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 268–284, Morgan Kaufmann, 1988.).

[22] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[23] Nicholas V. Findler and Ron Lo. An examination of distributed planning in the world of air traffic control. *Journal of Parallel and Distributed Computing*, 3:411–431, 1986.

[24] Mark S. Fox. Organization structuring: Designing large complex software. Technical Report 79-155, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, December 1979.

[25] Mark S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 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.).

[26] Jay Galbraith. *Designing Complex Organizations.* Addison-Wesley, 1973.

[27] Jay R. Galbraith. *Organization Design.* Addison-Wesley, 1977.

[28] L. Gasser. The integration of computing and routine work. *ACM Transactions on Office Information Systems*, 4(3):205–225, July 1986.

[29] Les Gasser, Carl Braganza, and Nava Herman. MACE: A flexible testbed for distributed AI research. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 5, pages 119–152. Pitman, 1987.

[30] Les Gasser and Nicolas Rouquette. Representing and using organizational knowledge in distributed AI systems. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[31] Michael Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 125–129, Washington, D.C., August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 200–204, Morgan Kaufmann, 1988.).

[32] Michael Georgeff. A theory of action for multiagent planning. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 121–125, Austin, Texas, August 1984. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 205–209, Morgan Kaufmann, 1988.).

[33] Michael Georgeff. A representation of events in multi-agent domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 70–75, Philadelphia, Pennsylvania, August 1986. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 210–215, Morgan Kaufmann, 1988.).

[34] S. Goyal and R. Worrest. Expert system applications to network management. In J. Leibowitz, editor, *Expert System Applications to Telecommunications*, volume 1, pages 3–44. John Wiley & Sons, 1988.

[35] 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, August 1985.

[36] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. In *Third ACM Conference on Principles of Distributed Computing*, 1984.

[37] Frederick Hayes-Roth, Lee D. Erman, Scott Fouse, Jay S. Lark, and James Davidson. ABE: A cooperative operation system and development environment. In Mark Richer, editor, *AI Tools and Techniques*. Ablex Publishing Corporation, 1988. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 457–489, Morgan Kaufmann, 1988.).

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

[39] Carl Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 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.).

[40] Michael Huhns, editor. *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.

[41] Michael N. Huhns, Uttam Mukhopadhyay, Larry M. Stephens, and Ronald D. Bonnell. DAI for document retrieval: The MINDS project. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 9, pages 249–284. Pitman, 1987.

[42] Kurt Konolige. Circumscriptive ignorance. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 202–204, Pittsburgh, Pennsylvania, August 1982.

[43] 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.

[44] William A. Kornfeld. ETHER: A parallel problem solving system. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 490–492, Cambridge, Massachusetts, August 1979.

[45] 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.).

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

[47] Amy L. Lansky and Dan Fogelsong. Localized representation and planning methods for parallel domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 240–245, Seattle, Washington, August 1987.

[48] Douglas B. Lenat. Beings: Knowledge as interacting experts. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 126–133, Stanford, California, August 1975. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 161–168, Morgan Kaufmann, 1988.).

[49] 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.

[50] Victor R. Lesser and Daniel D. Corkill. The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, Fall 1983. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 353–386, Addison-Wesley, 1988 and in *Readings from AI Magazine: Volumes 1–5*, Robert Engelmore, editor, pages 69–85, AAAI, Menlo Park, California, 1988).

[51] 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*, Alan H. Bond and Les Gasser, editors, pages 120–139, Morgan Kaufmann, 1988.).

[52] David Maines. Urban life, 1984. Special issue on negotiated order theory.

[53] T. W. Malone and S. A. Smith. Tradeoffs in designing organizations: Implications for new forms of human organizations and computer systems. Working Paper CISR WP 112 (Sloan WP 1541-84), Center for Information Systems Research, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, March 1984.

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

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

[56] C. Mason, R. Johnson, R. Searfus, D. Lager, and T. Canales. A seismic event analyzer for nuclear test ban treaty verification. In *Proceedings of the Third International Conference on Applications of Artificial Intelligence in Engineering*, August 1988.

[57] Murray S. Mazer. Exploring the use of distributed problem-solving in office support systems. In *Proceedings of the IEEE Computer Society Symposium on Office Automation*, pages 217–225, April 1987.

[58] James L. McClelland, David E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (2 Volumes).* MIT Press, 1987.

[59] Nils J. Nilsson. Two heads are better than one. *SIGART Newsletter*, (73):43, October 1980.

[60] Sergei Nirenburg and Victor Lesser. Providing intelligent assistance in distributed office environments. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 590–598. Morgan Kaufman, 1988.

[61] H. Van Dyke Parunak. Manufacturing experience with the contract net. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 10, pages 285–310. Pitman, 1987.

[62] H. Van Dyke Parunak, Bruce W. Irish, James Kindrick, and Peter W. Lozo. Fractal actors for distributed manufacturing control. In *Proceedings of the second IEEE Conference on AI Applications*, pages 653–660, December 1985.

[63] Scott Reed and Victor R. Lesser. Division of labor in honey bees and distributed focus of attention. Technical Report 80-17, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, September 1980.

[64] James P. Rice. Poligon: A system for parallel problem solving. Technical Report KSL-86-19, Knowledge Systems Laboratory, Computer Science Department, Stanford University, Palo Alto, California 94304, April 1986.

[65] Jeffery S. Rosenschein. Synchronization of multi-agent plans. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 115–119, Pittsburgh, Pennsylvania, August 1982. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 187–191, Morgan Kaufmann, 1988.).

[66] Jeffrey S. Rosenschein and Michael R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, Los Angeles, California, August 1985. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 227–234, Morgan Kaufmann, 1988.).

[67] Jeffrey S. Rosenschein and Michael R. Genesereth. Communication and cooperation among logic-based agents. In *Proceedings of the Sixth Phoenix Conference on Computers and Communications*, pages 594–600, Scottsdale, AZ, February 1987.

[68] Stan Rosenschein. Reasoning about distributed action. *SIGART Newsletter*, 84:7, 1983.

[69] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier, 1977.

[70] Arvind Sathi, Thomas E. Morton, and Steven F. Roth. Callisto: An intelligent project management system. *AI Magazine*, 7(5):34–52, 1986.

[71] Eric Schoen. The CAOS system. Technical Report STAN-CS-86-1125, Computer Science Department, Stanford University, Stanford, California 94305, March 1986.

[72] Herbert A. Simon. *Models of Man.* John Wiley & Sons, 1957.

[73] Herbert A. Simon. *The Sciences of the Artificial.* MIT Press, 1969.

[74] David Smith and Martin Broadwell. Plan coordination in support of expert systems. In *Proceedings of the DARPA Knowledge-based Planning Workshop*, Austin, Texas, December 1987.

[75] Reid G. Smith. A framework for distributed problem solving. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 836–841, Cambridge, Massachusetts, August 1979.

[76] 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.

[77] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):61–70, January 1981. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 61–70, Morgan Kaufmann, 1988.).

[78] Reid G. Smith and Randall Davis. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.

[79] Reid Garfield Smith. *A Framework for Problem Solving in a Distributed Processing Environment.* PhD thesis, Stanford University, December 1978. A revised version was published by UMI Research Press.

[80] Stephen F. Smith and Juha E. Hynynen. Integrated decentralization of production management: An approach for factory scheduling. In C. R. Liu, A. Requicha, and S. Chandrasekar, editors, *Intelligent and Integrated Manufacturing Analysis and Synthesis*, pages 427–439. The American Society of Mechanical Engineers, New York, 1987.

[81] John A. Stankovic, Krithivasan Ramamritham, and Sheng-Chang Cheng. Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. *IEEE Transactions on Computers*, C-34(12):1130=1143, December 1985.

[82] R. Steeb, S. Cammarata, S. Narain, J. Rothenburg, and W. Giarla. Cooperative intelligence for remotely piloted vehicle fleet control. Technical Report R-3408-ARPA, Rand Corporation, October 1986.

[83] A. Strauss. *Negotiations: Varieties, Processes, Contexts, and Social Order.* Jossey Bass, San Francisco, 1978.

[84] Katia Sycara. Planning for negotiation: A case-based approach. In *DARPA Knowledge-Based Planning Workshop*, pages 11.1–11.10, December 1987.

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

[86] Katia P. Sycara. Multi-agent compromise via negotiation. In *Proceedings of the 1988 Distributed AI Workshop*, May 1988.

[87] Katia Sycara-Cyranski. Arguments of persuasion in labour mediation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 294–296, Los Angeles, California, August 1985.

[88] Perry W. Thorndyke, David McArthur, and Stephanie Cammarata. Autopilot: A distributed planner for air fleet control. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 171–177, August 1981.

[89] Robert Wesson, Frederick Hayes-Roth, John W. Burge, Cathleen Statz, and Carl A Sunshine. Network structures for distributed situation assessment. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):5–23, January 1981. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 71–89, Morgan Kaufmann, 1988.).