ARPA Order No.: 3460/3473
0P10 Information Processing Techniques

R-2560-ARPA
August 1980

Network Structures for Distributed Situation Assessment

R. Wesson, F. Hayes-Roth

with J. Burge, C. Stasz, C. Sunshine

A Report prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under Contract No. MDA903-78-C-0029.

Published by The Rand Corporation

ARPA Order No.: 3460/3473
0P10 Information Processing Techniques

R-2560-ARPA
August 1980

Network Structures for Distributed Situation Assessment

R. Wesson, F. Hayes-Roth

with J. Burge, C. Stasz, C. Sunshine

A Report prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

PREFACE

The concept of a *distributed sensor network* (DSN) for automatic surveillance tasks is being investigated in studies conducted for the Defense Advanced Research Projects Agency. Rand's research effort has focused on characterizing, evaluating, and comparing alternative candidate DSN architectures, and has included a series of laboratory experiments in a simplified environment. This report documents these experiments and presents recommendations drawn from them that should be useful to other researchers in their efforts to develop DSNs. The research was sponsored by ARPA's Information Processing Techniques Office under Contract MDA903-78-C-0029.

SUMMARY

Increases in weapon counts, coupled with today's small weapon sizes, have made *situation assessment* (SA) increasingly difficult; moreover, current SA systems are highly centralized and vulnerable. Technological advances in cheap sensors, microprocessors, packet radio communications, and artificial intelligence offer promising alternatives for performing military surveillance in many situations. A new approach now being investigated is that of an automated *distributed sensor network* (DSN) consisting of many "intelligent" sensor devices that can pool their knowledge to achieve an accurate overall assessment of a situation.

We have conducted laboratory experiments to investigate potential DSN organizations and to ascertain some general design principles. These experiments have been performed with a laboratory task that we have called the *message puzzle task* (MPT). The MPT manifests the important information-processing characteristics of SA tasks, yet is simple enough for laboratory experiments. Using a real-time computer system, the MPT simulates a two-dimensional environment in which numerous entities move toward distributed targets. A network of "sensor nodes," each of which sees only a small portion of the entire environment, attempts to identify the mobile entities as quickly as possible. To do this, they must cooperatively communicate their hypotheses and data, using a limited number of messages.

We have tested two general DSN organizations, using the MPT. The first, called the *anarchic committee* (AC), consists of nine sensor nodes. Each node "sees" roughly one-ninth of the total environment. Any node can send messages to one, some, or all other nodes. The other organization, called the *dynamic hierarchical cone* (DHC), employs 13 nodes organized on three hierarchical levels. The lowest level has nine nodes that each see one-ninth of the environment, as in the AC organization, but they cannot communicate with each other. The next level has three "middle-managers," which cannot see any part of the environment directly and cannot communicate with each other, but which can communicate with the lower-level nodes. The one highest-level node can get data only from the middle level. Thus, communication occurs between but not within levels. Within these constraints, the nodes are free to organize and communicate in any way they find effective.

We conducted a series of experiments using Rand research personnel as nodes. In these experiments, the AC consistently outperformed the DHC. The AC identified more entities faster and more accurately; it used far fewer messages; it shared both low- and high-level data faster than the DHC; and it formed more and better abstract hypotheses than the DHC. The DHC rarely developed global assessments, apparently because its middle-level nodes were too heavily loaded to perform integrative functions effectively. Only by using middle-level nodes as intermediaries could the lower-level nodes share raw data; that task required virtually all of the middle-level nodes' resources and forced them to adopt message-packing and filtering techniques.

These experiments support our contention that DSN architectures need to emphasize cooperative aspects of problem-solving, rather than more familiar issues such as problem-reduction and subgoaling. In particular, designers should pay

special attention to the problems of increased overhead, overloading, and inappropriate information flow which characterize strict hierarchical arrangements. Domain and network characteristics both determine the extent to which processing, authority, and communication should be hierarchically organized. In the DSN, the nodes seem able to process very low-level signals with very little cooperation. Results can be reported directly upward to integration nodes. Functions at these higher-level nodes, where most of the "intelligence" of the network lies, should not be strictly partitioned. This level requires lateral communication to contrast mutually competitive hypotheses and to integrate mutually supportive ones.

One candidate design for minimizing redundant communication under such conditions uses model-based reasoning to form expectations that guide, limit, and reduce reporting frequency. The network, over time, evolves a set of mutually supportive hypotheses constituting a model of the sensed world. We constructed and tested a simulation of such a network on Rand's PDP 11/70 computer system. Not only did we demonstrate the efficacy of such a DSN design, we were also able to achieve performance levels comparable to those of the human AC. The machine network, moreover, required far fewer messages than the human AC. Nevertheless, it still required more messages than would be ideal in an actual SA task.

Because most of the communications in such a network concern hypothesis updating and revision, we suggest a new method for representing hypotheses to minimize communication requirements. This concept is called the *process assembly network* (PAN). A PAN replaces traditionally passive data hypotheses by active "hypothesis processes" that are responsible for predicting their own evolution over time. These hypotheses enable nodes to represent and predict the "belief systems" of their relevant neighbors, thus supplanting the typically frequent data-reporting task with a system for "management by exception," where only surprises need to be explicitly communicated.

ACKNOWLEDGMENTS

Rand staff members provided valuable assistance during this project. Norman Shapiro provided helpful insights when we first grappled with the task of characterizing distributed situation assessment. We must also thank the volunteers who formed the pool of human DSN nodes, as their participation assured us of first-hand experience with the subject of this report: Patricia Aicher, Robert Anderson, Monti Callero, Stephanie Cammarata, Nancy Daniel, William Faught, R. Stockton Gaines, William Giarla, Dan Gorlin, Barbara Hayes-Roth, Philip Klahr, Walter Matyskiela, Doris McClure, Ralph Strauch, and Perry Thorndyke.

CONTENTS

FIGURES

I. INTRODUCTION

Situation assessment (SA) involves acquiring, organizing, and abstracting information about the environment which may either correlate well with expectations or serve to create new ones. Large-scale SA usually involves a network of information producers and consumers which handle information that varies by locale, amount of aggregation, and level of abstraction. Timely SAs in current military environments require systems that can process more of this information faster than has ever been done before.

New information-processing technologies suggest that more of the SA task can be automated. Advances in microprocessors, packet-switching radio communications, sensors, and artificial intelligence methods for automated problem-solving all combine to form the foundations of a highly automated, low-cost, intelligent, *distributed sensor network* (DSN). A DSN consisting of inexpensive, intelligent nodes scattered across a battlefield would be useful in acquiring noisy intelligence data, processing it to reduce uncertainty and produce higher-level interpretations, and transmitting this abstracted information quickly to the commanders who require it (see Fig. 1). The development of DSNs is proceeding rapidly, both for this problem area [1, 2] and other related areas, including traffic control [3] and industrial control [4].

The research reported here addresses the basic question, What computer network organizational structures are best suited to the SA task? In attempting to characterize suitable structures for a DSN, we have been influenced by several

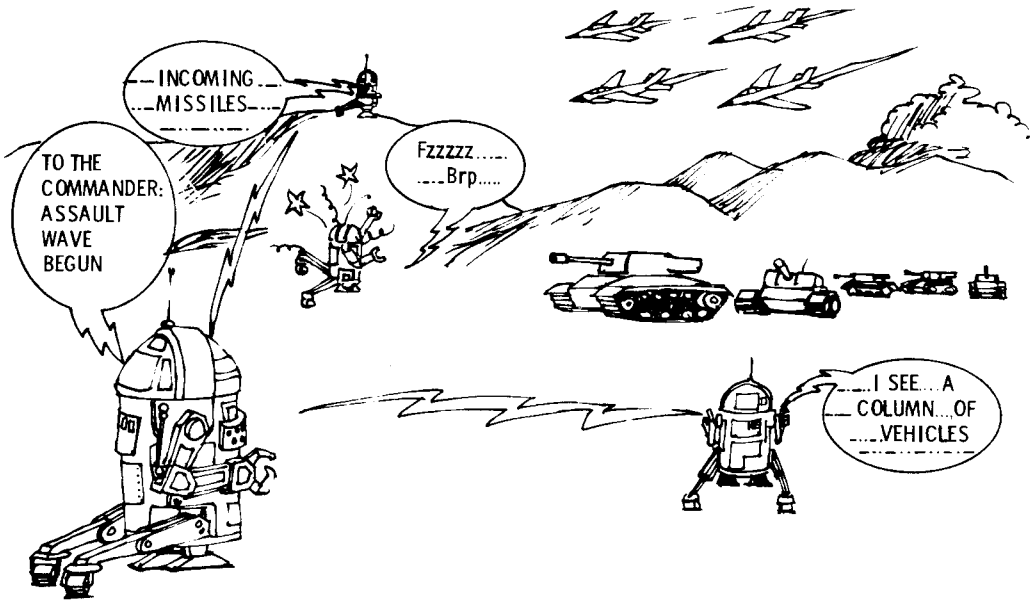

Fig. 1—The distributed sensor network

general hardware trends: Processing power is increasing at a rapid rate while hardware costs are decreasing at about the same rate; communication capabilities are also becoming better and less costly, although significantly more slowly than computational power; low-cost and low-power sensors are likely to remain moderately unreliable. We have attempted to keep our research sensitive to the foregoing relationships among DSN elements while avoiding restrictions imposed by specific current hardware designs and constraints.

Two structures seem appropriate for SA tasks [5]. The first, a committee structure called the *anarchic committee* (AC), facilitates communication and reorganization; the second, called the *dynamic hierarchical cone* (DHC), presumably provides a superior global problem perspective. To compare these structures, we chose a simplified SA task. This experimental task, which we have called the *message puzzle task* (MPT), was created by converting many of the elements of a battlefield SA task into a more familiar information-processing problem. In it, messages (words and phrases) move in a snake-like manner through a field sparsely littered with obstructions before stopping at their final destinations. The words represent moving platforms that emit spectral signals. The task is to identify incoming platforms as quickly as possible before they reach their destinations. Each member of the organization is limited in his field of view, so cooperation is imperative for the organization to perform its mission well.

In these experiments, the AC consistently outperformed the DHC. Our analysis of these results leads us to believe that DSN designers should emphasize cooperative aspects of problem-solving rather than issues such as subgoaling or problem-reduction. Communication requirements can be reduced if passive data hypotheses are replaced by active hypothesis processes that predict their own evolution over time. We therefore designed a *process assembly network* (PAN) that enables nodes to represent and predict the "belief systems" of their relevant neighbors. With a PAN, frequent data-reporting is not necessary—only the unexpected must be reported.

We have performed both man-machine and machine-only experiments to compare alternative organizations. The next section describes in detail the laboratory task and the two organizations tested, as well as the experiments and their results. Integrated with previous work from organization theory, our laboratory experiences provided useful insights into DSN designs, which are explicated in general terms in Sec. III. Then, Sec. IV discusses pure machine-based DSN structures. A set of experiments are described in which a computer program performed the MPT, and better and more complex designs for eventual implementation are suggested. Finally, some "heuristics of cooperation" that we identified during our studies are given in the Appendix.

II. BRINGING DISTRIBUTED SITUATION ASSESSMENT INTO THE LABORATORY

The question, Which organizational structures are best suited to SA tasks?, is too general and abstract to address directly. We therefore began by attempting to clarify the notion of "organizational structure." We addressed the following broad questions:

- *Control hierarchies:* How many levels of managerial authority should the SA structure have?
- *Hypothesis formulation and sharing:* Who should construct the more abstract or more aggregated world views? How is formulation of high-level interpretations related to structure? Who should communicate them out of the network?
- *Communication:* What is the most efficient communication network structure? Who needs to talk to whom?
- *Raw data acquisition and sharing:* Should the nodes that acquire raw data send it to a higher level for further processing or deal with it directly?
- *Adaptation to environmental changes:* Which structures most readily adjust to rapid external changes? Within a structure, what constitutes an effective change?

Next, we had to select a representative SA task that was suitable for laboratory experimentation. The problem had to be one in which confusing, perishable, and componential information could be aggregated to interpret and explain high-level events. Military battlefield environments, for example, are too complex and ill-defined to support this type of laboratory analysis, so we had to construct a reduced domain that captured the essence of the generic case. Our requirements for this task included:

- A battlefield-like scenario consisting of objects moving around both individually and in formation; typical kinds of actions, such as concealment and cover, coordinated attacks, and feints.
- Multiple sensors, each of which can see some, but not all, of the low-level activities; reported data that contain errors and permit incompleteness.
- Perishable data, in the sense that information value depends critically on timely processing.
- Limited communication among processors, to permit investigation of the communication-computation tradeoffs.
- A rich data environment containing enough information to allow the system to overcome the above limitations and achieve a solution.

We designed an experimental task manifesting these characteristics, which then formed the basis for all of our experimental work.

THE MESSAGE PUZZLE TASK

The task designed for the experiments, the MPT, is a game-like environment in which multiple players cooperate. We called it the *message puzzle task* because it involves messages consisting of words and phrases that move about in a two-dimensional grid like that of a puzzle board. A group of players, each of whom can see only a small portion of the grid, must communicate among themselves and identify the moving items as quickly and accurately as possible. The MPT configuration is shown in Fig. 2. All of our experimental tasks used 15×15 puzzle boards, divided into nine overlapping views. Each view measured approximately 6×6 squares in size.

This task, although apparently quite simple, requires very sophisticated information-processing capabilities and cooperative strategies. To interpret the data each had in his own view, along with other data received over the message channels, the players had to:

- Auto-correlate successive sensor reports, using past letter and word motions to achieve a current understanding of the contents of their portion of the grid.
- Filter out noise and modify their reports of raw data accordingly.
- Infer or detect terrain features (black squares blocking motion).
- Send or request raw data such as complete views, the locations of black squares, and the like.
- Hypothesize and extrapolate tracks or unit identities to match their own sensor reports with those of adjacent nodes.
- Send or request hypotheses of the above.

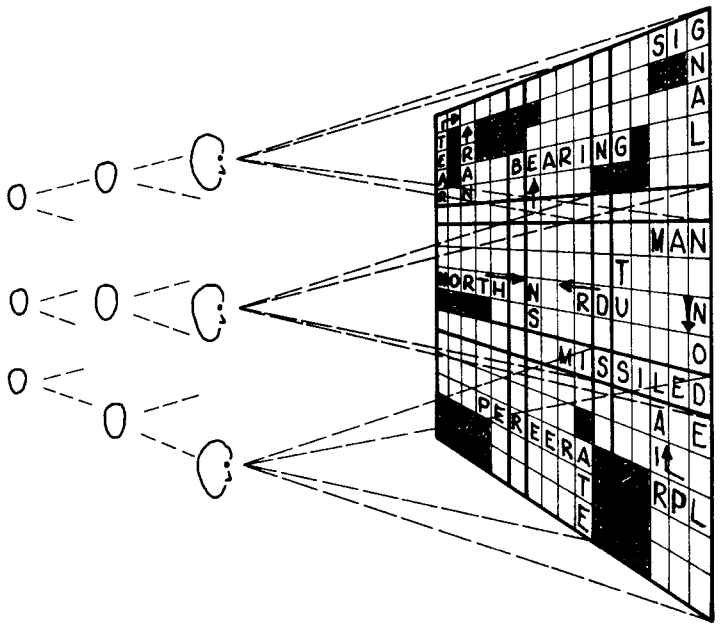

Fig. 2—The message puzzle task (MPT)

- Maintain a time history of hypotheses in order to assist future hypothesis generation.
- Predict future behaviors of elements.
- Hand off tracking, hypothesizing, and guessing responsibilities to other nodes.
- Assign or request processing resources and task responsibilities.

The MPT has the most important characteristics of typical SA tasks: Information must be processed at multiple levels of abstraction; elements in the world are constantly moving or changing; no single individual can solve the task alone, so cooperation is required for success; the many constraints present in the data and the motion of the data allow effective processing in the context of perishable and errorful data; and the communication channels between individual nodes are severely limited relative to the processing capability of each node.

ORGANIZATIONS TESTED

Within this task environment, we tested two very different organizational structures as candidate DSN systems. The first organization is a "flat," non-hierarchical structure known in the artificial intelligence (AI) literature as the "cooperating experts" paradigm.* In organizational theory, such structures are denoted as "Type X" organizations [7]; examples include crisis-management teams and small R&D firms with loose management styles. In general, organizations of "cooperating experts" are composed of specialists and have little or no hierarchical structure. Problems are solved by sharing individual perspectives, which refine and ultimately integrate local interpretations into a unified group consensus. Subtasking, reporting requirements, and resource-allocation decisions are generally not specified a priori. The organization forms behavior and communication patterns dynamically in response to the environment and changes the patterns in a data-directed way.

In direct contrast to the "cooperating experts" paradigm are the very hierarchical "Theory Y" or "perceptual cone" organizations. Organizational theory refers to large, established steady-state firms as an example of Theory Y organizations [7]. "Perceptual cone" is a term coined in early AI work in pattern recognition [8]. Organizations of this class are assembled as strict hierarchies of abstraction levels. At each level, individual elements receive reports from the levels below them, integrate the reports according to their special skills and position in the hierarchy, and report upward the abstracted versions of their results. The highest level of the network may repeatedly order its subordinates to adjust some previous reports in accordance with its own global perspectives, or it may report the overall interpretation it has formed. As each node has precisely specified input/output (I/O) activity and task requirements, this type of organization is generally found in domains requiring routine, but complex, information processing. (The "cooperating experts" organizations tend to be favored in less complex, but more uncertain or rapidly changing, environments.)

The MPT domain involves both types of information processing. Letters must routinely be tracked and combined together into fragments which must then be

*Perhaps the best-known AI work of this type is exemplified by the Hearsay-II speech understanding system [6].

used to generate word and phrasal guesses. At the same time, it is a highly uncertain and rapidly changing environment in which noisy data and node failures occur unpredictably.

Since the MPT, like the full SA task, exhibits different features which suggest that each organizational structure might be suitable, we tested both. We called the first structure the *anarchic committee* (AC) to reflect the absence of overt governmental structure and the tendency to spawn many overlapping committees to perform specific tasks. The second structure is of the form of a *dynamic hierarchical cone* (DHC)—a "perceptual cone" organization, modified to be more responsive to either a spatially unbalanced or rapidly changing data flow. Both organizations have been described in earlier work [5], so our treatment here will be brief.

The AC consists of the nine nodes shown in Fig. 2 plus a fully interconnected communications network (see Fig. 3). The system is designed to exhibit minimal organizational constraints. Message passing, for example, is simple, since complete simultaneous broadcasting is provided. Each node can communicate with any other node. Messages sent to "all" are delivered to all nine nodes. Additionally, the structure imposes no natural authority decomposition. We assumed that cooperation would occur via the formation of geographic committees sharing data on particular words and phrases. Coordination between these mutually supportive committees was expected to be limited and, at best, ad hoc.

The DHC, on the other hand, is expected to coordinate its elements much more easily, by virtue of its structure. As shown in Fig. 4, two levels of "manager" nodes are added to the basic nine positions of the AC. Conceptually, above the nine low-level nodes are three middle-level nodes. Above the middle level is a single high-level node. Communication is restricted to flow between adjacent layers; there is no communication within a layer. Each of the nine low-level nodes has the same

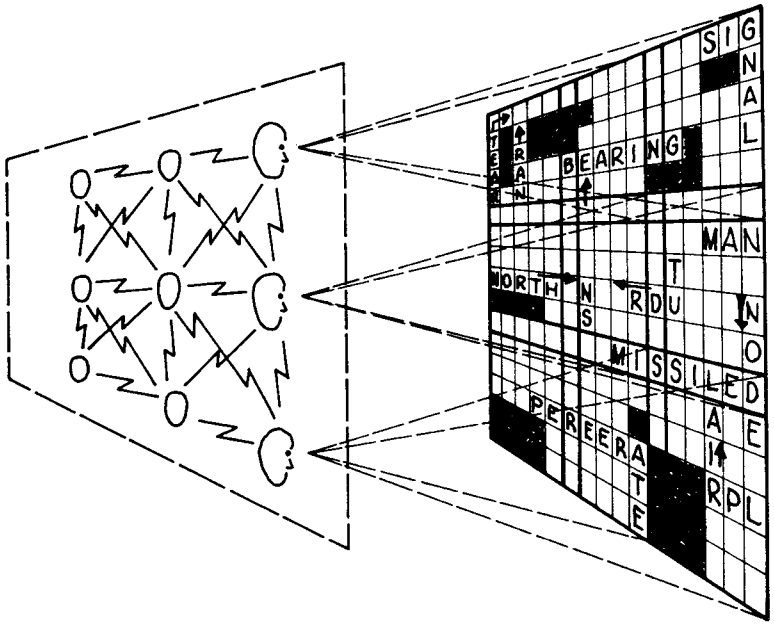

Fig. 3—The anarchic committee (AC)

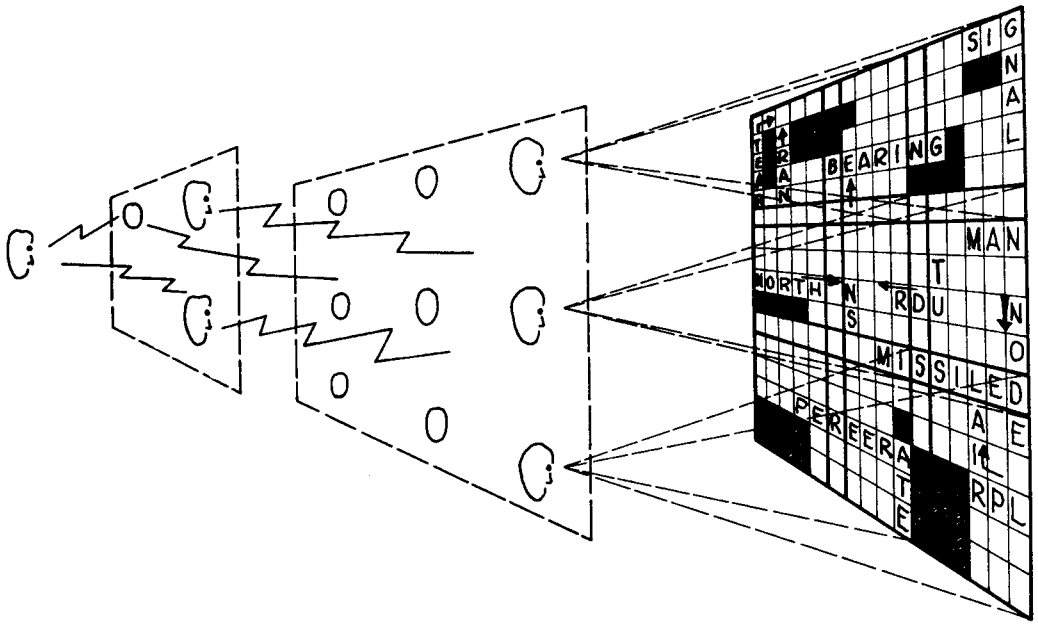

Fig. 4—The dynamic hierarchical cone (DHC)

view of the puzzle as the nodes in the AC, but the DHC nodes can communicate directly only with the three middle-level nodes. The single top-level node can communicate only with the middle levels. All four higher-level nodes receive no data reports of their own but must rely on reports from low-level nodes.

There is a natural allocation of authority in this structure. Middle-level nodes are expected to function as "middle managers," carrying out the instructions of the high-level node, which presumably would acquire the most global perspective. The low-level nodes are expected to function as intelligent sensors, preprocessing data for subsequent integration by their middle-level managers. The assignment of low-level nodes to middle-level managers was left open in our experiment, allowing the managers to establish whatever scope of responsibility and reporting structures seemed appropriate and to reconfigure them as events unfolded.

Communication restrictions were imposed on both organizations. Each node could generate a prespecified number of messages per unit time. Message reception and processing were limited only by the nodes' abilities to process the incoming data. Other than these, no a priori restrictions were placed on nodes' utilization of communication resources.

HUMAN-BASED NETWORK EXPERIMENTS

Apparatus, Subjects, and Procedure

A major goal of the initial experiments, in which human players acted as nodes, was to capture information in a form suitable for analysis. We therefore attempted

to develop a computer-based testbed that would provide better tools with which each node could perform its task and at the same time would record traces and performance data automatically. In the resulting system, each node is provided with a CRT terminal for sending messages to other nodes (within the constraints of the problem), making official reports of puzzle entities to an outside "controller," and displaying its current view of the grid (if it has one).

Using this testbed and the organizations described above, we conducted a series of informal experiments. These experiments were not designed with strict experimental controls to permit formal hypothesis testing. Rather, they were undertaken to develop familiarity with the limitations imposed by each of the organizational structures and to derive heuristics for use by a computer program that would support properly controlled experiments. Six separate trials or "problems" were run. Three trials employed the AC organizational structure, and three employed the DHC structure. A different puzzle was used in each case.

Prior to each trial, subjects met to receive node assignments (i.e., node positions) and to develop standard problem-solving strategies and procedures. These standards covered both low-level conventions for formatting messages (e.g., "in messages, indicate that a word was already officially guessed by marking it 'g!'") and higher-level strategies (e.g., "to reduce repetitious official guesses, send formal hypotheses for external communication to the high-level node who alone can report them"). Nodes also exchanged information about how they approached their individual problem-solving tasks, such as methods for tracking word or letter motions. These planning meetings lasted about 30 minutes. Subjects then performed the task. Total time for each problem was about 1 hour. A short debriefing followed, in which subjects provided general impressions of the exercise, their problem-solving heuristics, and new ideas for improving procedures in the future. They also completed a short post-experiment questionnaire.

Experimental Analysis and Results

The first two problem-solving sessions were treated as practice sessions to familiarize the participants with each organizational structure. The last four sessions were analyzed and compared, first in terms of absolute problem-solving performance (accuracy and timeliness of guesses) and later in more detailed ways.

Since each problem used a different puzzle, we could not assess the reliability of observed differences between problems. Differences could be due to puzzle difficulty and node variability as well as to organizational structure. Recognizing this limitation, it seemed reasonable to compare actual node performance with some global performance criterion for that particular problem. Therefore, for each problem, we computed benchmark performance by a hypothetical "global node." By assumption, a global node could see the whole puzzle grid without errors and had a list of rules or heuristics for formulating hypotheses and official reports. For each successive round (one data update every few minutes), a scorer applied these rules to the current view to determine which words the global node would report.

Performance results are presented in Fig. 5. Each graph displays the cumulative percent of correct reports by the global and actual nodes at each round. The shaded area between the curves indicates the extent to which actual performance falls short of the global criterion. Overall, the AC organization appeared to perform

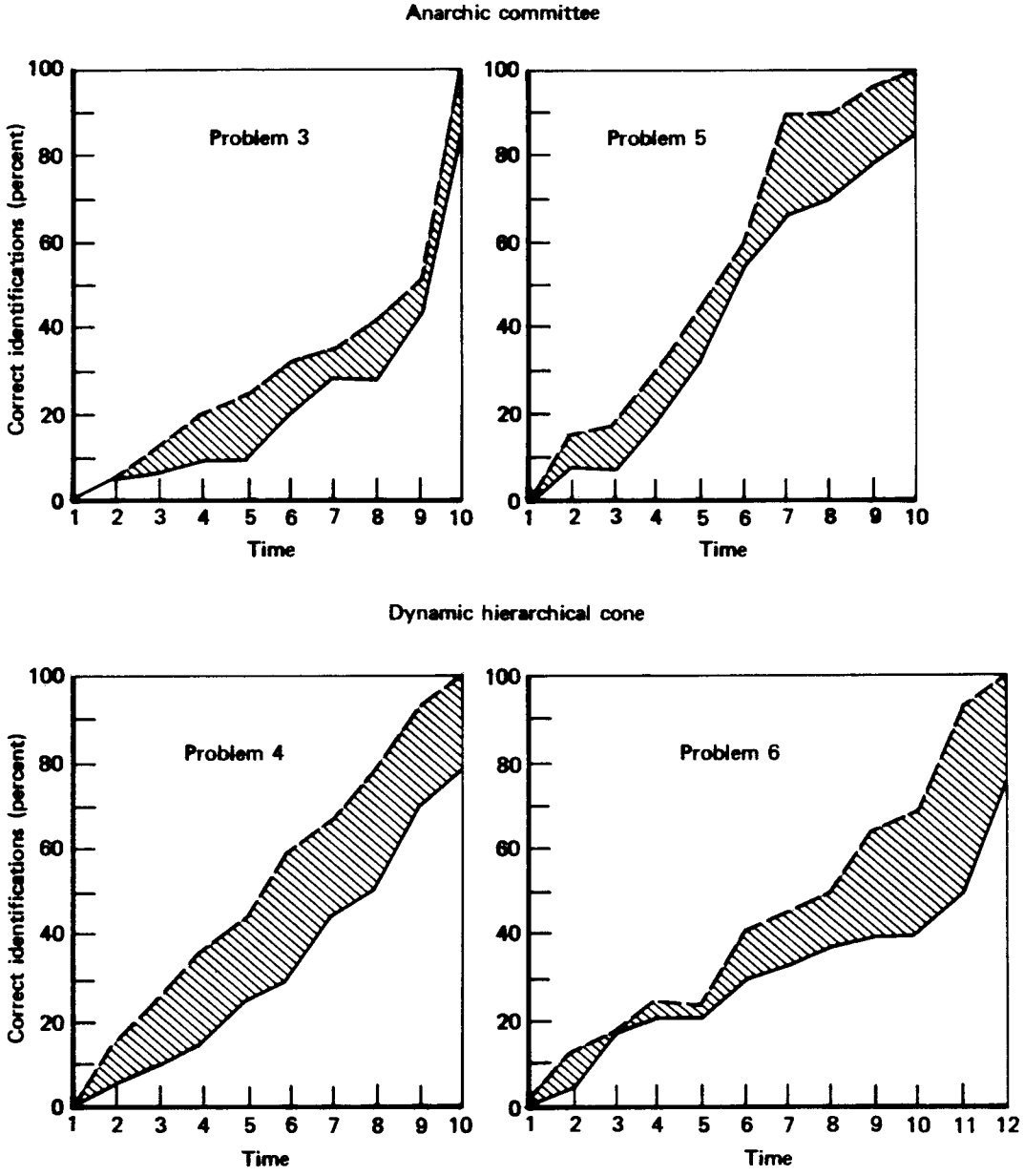

Fig. 5—Overall performance: Problems 3–6

better than the DHC, with AC performance closer to the global benchmark reporting levels in all cases.

To study the organizational dynamics that lead to good performance, we conducted further analyses of the problem-solving traces. These analyses focused on three general conjectures:

1. The AC communicated more efficiently than the DHC, using fewer messages to achieve comparable performance.
2. The AC promoted faster communication than DHC, with nodes responding to requests for information more quickly.
3. Higher-level, more abstract information—such as phrase and theme hypotheses—was shared and understood more often in the AC structure.

The first conjecture was clearly true: DHC teams used about twice as many messages per problem as AC teams. This happened primarily because of differences in self-monitoring and task apportionment communications. The authority structure in the DHC stimulated many more reports and requests about node activity (e.g., "busy or idle now?"). Upper-level managers used this information to redistribute task responsibilities, such as assigning lower-level nodes to managers. The basic communication constraints of the DHC also generated many more messages because of the need for messages to be relayed through the hierarchy. Messages were also repeated much more frequently in the DHC. There were three times as many repeats in DHC problems (21 repetitions) as in AC problems (7 repetitions), indicating widespread dissatisfaction with the greater delays in DHC processing (see below).

This leads to the second conjecture—that the AC communicated information faster. In the dynamic MPT situation, speed of communication is essential. Since words overlap viewing areas and views change with every round, information in this task is highly perishable. By the time the information is received, it may be obsolete. On the average, the AC response time was about 1 minute, while the DHC response time was 3 to 4 minutes. We found that response delay for middle-level managers was much higher than that for low-level nodes, reflecting the difficulty of the managers' task. Middle-level managers receive requests for information from many independent nodes. If they know the information, they can respond immediately. If not, they must make inquiries and wait for replies before they can respond to the original message. They must also remember who requested which information. Thus, bottlenecks at the middle level caused significant delays in information exchange.

The third conjecture concerns levels of abstraction in the MPT. Higher-level solutions, such as phrases and themes, are more difficult than simple single-word identifications. Since phrases overlap between areas, a single node can never see a complete phrase. Therefore, higher-level solutions require a more global perspective of the puzzle than is obtained by viewing a single area. The AC identified 83 percent of the possible phrases, while the DHC identified only 62 percent. Furthermore, the AC consistently identified phrases earlier than the DHC: The average time beyond the global-node criterion for reporting phrases was 1.2 rounds in the AC, as compared to 2.6 rounds in the DHC. Finally, while the AC identified all but one phrase before the final round, the DHC did not identify any phrases until the final round.

These results thus tend to support the following general conclusion: In the MPT domain, the committee structure performs best, reporting more information faster, using fewer communicating resources.

III. IMPROVED CONJECTURES ON DISTRIBUTED ARTIFICIAL INTELLIGENCE

The fact that the AC performed so consistently better than the DHC indicates that the AC structure manifests some especially appropriate properties for the SA task, or that the DHC structure has some especially bad properties for it, or perhaps both. We think the third possibility is most probable. We wish to emphasize that *we do not think that committees should replace existing hierarchical organizations, nor do we believe that all networks should have fully connected broadcast communication channels.* Our work is too limited to permit that sort of generalization. We do think, however, that we have explored a relevant portion of the range of DSN design choices and that our results, suitably aided by our now improved conjectures, can provide a valuable basis for improved DSN designs.

TASK DECOMPOSITION

How should an organization subdivide SA problems? Our best recommendation remains the same as the hypothesis with which we began this research: Activities should be divided among multiple cooperating knowledge sources or specialists producing competing and ultimately cooperating hypotheses. Since communication constraints limit and delay data dissemination, and processing constraints limit the effectiveness of centralized problem-solving, decomposing the overall SA task into pieces assignable to the distributed elements seems most desirable. Assuming that we know the *knowledge source* set necessary to solve this problem centrally, the question then becomes, How can we endow a network of processing elements with the initial specialists, interconnections, and additional specialists required for data- and hypothesis-sharing to achieve good overall problem-solving performance?

Natural geographic boundaries (such as sensor or communication limits) provide one way to organize processors. However, geographic separation necessarily limits the system's capability to achieve global coordination of information gathering, organizing, and reduction activities. The tradeoff here is between rapid, local, data-directed processing and (usually less responsive) global, goal-directed processing. Traditional wisdom suggests that complex low-level data processing, such as the conversion of raw signal spectral data into power spectra, should occur near the sensor reporting the data, while higher-level inferences, such as platform identification, may require a more global perspective utilizing multiple sensor data. The low-level processing should be geographically assigned to sensor/low-level processor nodes; the higher-level inferencing may be released from this proximity-based restriction if communications are sufficiently accurate and swift.

If there are no geographical constraints, the choice between a "cooperating experts" and a hierarchical approach becomes clearer. Because of its sheer size, of course, a DSN will undoubtedly require some hierarchical levels of processing abstraction to coordinate information gathering, inference, hypothesizing, and control. However, we feel that since the upper levels must frequently communicate with the lower ones, and since this may be difficult because of differences in types

of data and processing for each, the number of such levels that are physically distinguished should be minimized.

Our experiments lacked the detail necessary to contribute much to the question of how many levels a DSN should contain. The MPT focused mainly on two abstraction levels—letters and words. Phrases and themes were infrequent and occasionally obvious in the context of a few words.

But we did demonstrate quite vividly how *not* to create a task hierarchy. Levels in hierarchies should be formed when global coordination can be improved by using data abstracted from a lower level. The essential requirement is that either the abstracted data must be reduced in complexity or the level being created must be given additional resources sufficient for the integration task. Neither condition existed in our MPT, and the middle-level nodes dealt with information too complex for the tools they had. On the other hand, many environments in which distributed organizations operate do exhibit the property of "natural abstraction levels." For example, the signal-processing environment seems to. The common abstractions found useful in other AI-based signal-processing tasks seem intuitive and appropriate: signal → spectral line groups → platforms and tracks → groups. Yet, even there, in the higher levels of abstraction (e.g., platforms), very low-level and complex data relationships (such as consistencies between the presence and absence of specific spectral lines) may depress the complexity-reducing effect of the abstraction process.

INTEGRATION

Integration denotes the degree and type of internode coupling required to solve a decomposed SA task. What information must nodes share? How often must they communicate? How can they reduce integration requirements to increase their own autonomy and thus decrease the waiting caused by subtask interdependencies? In a well-integrated organization, subtasking occurs smoothly with a minimum of communication.

One method of achieving integration is through the use of *protocols*. Communication protocols ("If I don't report a platform's position, that means I don't know it . . ."), authority protocols (". . . so don't ask me for it."), tasking protocols ("When you see the platform I've been telling you about, take over tracking and notify me.")—all of these can reduce the need for coordination during the problem-solving process itself. They do require extensive coordination at some point, though, either by system designers during conception or during periods of network inactivity. Furthermore, if critical data are filtered by the use of an inappropriate protocol, performance may suffer, with nobody being aware of any difficulties. Thus, while protocols serve a valuable function by encoding routine interactions, they must allow flexibility for rapidly changing environments.

A more important integration technique is the use of *expectations* to reduce communication and coordination needs. Human behavior exemplifies this technique. Particularly in specialty domains, linguistic patterns evolve that support tersely informative exchanges—witness the cryptic voice communication between pilots, which can usually be understood only by the participants. Linguistic expectations provide a basis for detailed models of communication partners.

The system designer must know how to generate, encode, maintain, and verify such assumptions, expectations, and predictions. Designing a DSN node that will model its neighbors in comparable ways requires knowledge about their processing states, pending hypotheses, incoming data that may be useful to itself, etc. In the absence of full communication, each node might contain executable simulations of neighboring processors, so that what communication does exist can be used to generate and maintain a consistent picture of what the neighbor might know. The cost-effectiveness of maintaining such a simulation of neighboring belief systems depends on the relationship between the parameters of communication, such as cost and availability, and those of computation, such as the ability to update and store alternate world states as often as necessary.

Even with modeling, communication is expected to be quite precious in a DSN. Maximizing the usefulness of shared data is of utmost importance. Hence, the question of *what* to communicate when candidate hypotheses or data items exceed channel capacity is a primary concern. Sharing a certain but obvious hypothesis is fruitless; likewise, telling a neighbor something you have little faith in may lead to unproductive inferences or effective deception. Nodes must evaluate how their hypotheses could benefit or confuse their collaborators to decide whether or not to communicate them. With limited communication, this evaluation may be largely guesswork. It can be improved significantly by maintaining predictive models of neighboring nodes, as mentioned above. But even then, uncertainty about how valuable a particular hypothesis is likely to be to neighbors, coupled with instances in which poorly supported hypotheses may influence key processing decisions for neighboring nodes, confuse this issue. Communications thus are always dependent on specifics of the dynamic environment.

AUTHORITY

Subtasking and integration require distributed authority in the network. Which nodes can perform task assignment? How many nodes should be controlled by one higher-level node? How centralized should decisionmaking be?

An immediate conclusion from the MPT experiments is that while centralized authority in the AC was absent by design, it also turned out to be absent in the DHC. Middle-level nodes, because of their processing and communications overload, were unable to exercise their implicit authority and ineffectively delegated it downward. For the same reasons, they were unable to provide the higher-level node with enough timely and correctly aggregated information to support his exercise of global judgment. Therefore, another conclusion (relevant once again to human organizations) is that nodes exercising any control must be given enough excess information-processing and communication resources (i.e., enough power) to wield their authority successfully.

Monitoring is an important function of authority. However, the implicit authority structure of the DHC contributed to its problem with excessive self-monitoring. Human experience verifies that, generally speaking, as the number of levels and individuals in authority increases, so does internal monitoring. To a certain point, this is good, since it supports a network's global perspective and overall coordination. Beyond that point, however, it simply consumes resources while dragging

down overall performance. The line between useful and excessive control messages in a dynamic distributed intelligence network is not precise. Some functions are clearly valuable—knowing if neighboring nodes have failed, for example, is essential for proper control of both global and individual behaviors. Other communications may be useful in some contexts but simply excess baggage in others. The contract-net concept of Smith and Davis [2] may provide a valuable framework for the dynamic assignment and exercise of authority, but when small, simple tasks are being negotiated, the administrative overhead of proposal, bid, acceptance, and assignment may prove excessively costly.

Another important duty of authority in a network is that of making optimal use of resources. This may mean reassigning tasks from overloaded nodes to less inundated ones. It may mean monitoring the problem-solving activity to detect nodes that are pursuing local hypotheses known from a more global viewpoint to be false. It may mean resolving conflicts, both of resources and of inferences. In order to perform this overseer role, a higher-level node must receive regular information from below. If the higher-level node is also integrating individual lower-level reports, those reports must not only communicate abstracted data but also process state information. Accomplishing this cheaply may require regular status reports.

A final note on authority: A network with a committee structure similar to our AC configuration seems to require very little authority assignment. Much, if not most, problem-solving is dictated by localized conditions. Nodes tend to correlate incoming data autonomously, communicating across sensor boundaries to neighbors to resolve ambiguity. Relatively few nodes must cooperate to achieve detection, classification, and tracking of targets. Many complex networks share these characteristics (e.g., the DABS sensor net [9] and the entire modern air traffic control network). It may be unnecessary to make authority explicit in a network that has sufficient resources at each node to obviate the need for extensive cooperation. If that is true, the real question for the design of a DSN is that of whether hardware costs are sufficiently low that increasing the number of nodes is a more cost-effective solution than the development of intelligent, cooperative software.

COMMUNICATION PATHWAYS

The conclusions discussed earlier concerning network communications—that hypotheses expected to be especially useful to others should be transmitted, that their transmission should occur at abstraction levels that reduce actual data transfer to a minimum, and that higher-level nodes will probably require regular reports from their subordinates for resource allocation and contention-resolving purposes —relate mainly to the "what" of communication. In the following, we address the "how" and "to whom."

Channel Capacity

In our MPT, the primary determinant of performance differences was the structure of communication partners. Members of the DHC, especially the low-level ones, were not allowed to communicate with their neighbors who had access to the essential data. They could share through intermediaries, but this was clearly a poor substitute. The middle-level nodes, moreover, were inundated by communications

requests they had neither the channel capacity nor the time to handle. And the highest-level node received too little information to be of service to the network at all. All of these problems were caused by inappropriate communications pathways.

The most serious deficiency was the limitation on low-level data-sharing. Abstracting data in this domain yielded little compression or reduction; too much higher-level processing depended on cues in the raw data. This effect may occur in many other SA domains as well.* We recommend that data-sharing among low-level nodes doing initial signal processing and cross-correlation be facilitated, allowing especially for the upward transfer of those data that are expected to be highly diagnostic.**

Allowing direct data-sharing at the lowest level would have greatly improved DHC performance. Giving the middle-level nodes additional channel capacity to handle the incoming data flows and attendant outgoing report requirements would likewise have improved their performance. Levels of hierarchy in a network typically represent a compromise between too much centralization (the extreme being centralized problem-solving with multiple sensors feeding a single integrator) and not enough (so that integration nodes have too little scope or support from below to form a useful global perspective). Whatever the choice for a node's span of purview and control, the node must have the communication capacity necessary to carry out its designed function.

Conversely, a fixed capacity constrains the feasible span of control for DSN hierarchies. That capacity may vary greatly across levels in the network, affecting the individual choices for processor and communication channel design in a variety of ways. One network design may stress uniformity of hardware to facilitate adaptivity to change. In a uniform system, each node would be a generalist with all the capabilities of another node. Replacement of failed nodes, for example, would be simplified, and network interconnections themselves would be standardized. However, this design must incorporate maximum expected channel capacity everywhere to preclude communication bottlenecks. This restriction is mitigated by natural levels of knowledge abstraction in many domains. In such hierarchical systems, moving up a level simultaneously reduces information complexity and increases its scope. This can result in a constant channel capacity requirement throughout the network.

Network Interconnections

Situation assessment tasks typically require localized coordination among neighbors with shared boundaries, and less frequent communication with distant nodes. (This was certainly true in our laboratory model.) The network design for communications interconnections should exploit this fact.

The need for longer-range data sharing increases with data abstraction. However, direct physical connections between all nodes that share such hypotheses are not necessarily required. Virtual connections such as those provided by packet

*As a vivid example, during the Mayaguez incident, the President received first-hand reports directly from observers at the scene of the action.

**This term was used by Hayes-Roth [10] to describe information that greatly limits the number of interpretations or hypotheses that are consistent with it. A diagnostic spectral line, for example, is one which uniquely determines a particular platform type. Its occurrence with a reasonable certainty allows all competing platform hypotheses and their attendant emitter and spectral signature interpretations to be immediatly dropped.

radio switching networks [11] could suffice, and they have the additional benefit of automatic adaptivity to changes in physical communication pathways. Broadcasting, either directly or via store-and-forward, would also work, provided each node can simply and quickly discriminate between messages it can use and messages to be ignored or forwarded. This is a complex issue, since the usefulness of information in highly dynamic environments cannot generally be determined at the time the information is generated or transmitted. Heuristics found useful in the MPT environment include ignoring messages from far away; stacking messages of a general nature behind those requesting immediate, simple, and precise action; and ignoring messages until local autocorrelation and hypothesis formulation are completed. (See the Appendix for these heuristics and more.)

Communication vs. Computation

Should nodes share information frequently and extensively, or should they reserve their communication resources while exploiting the data on hand? This is a classic tradeoff, and each approach involves some risk: Not communicating may result in the generation of much analysis that could be obviated by the diagnostic data conveniently available elsewhere; communicating too much may clog channels and waste time. How should a node decide what to do?

One possible solution would use a little communication to save a lot of computing: When starting a complex computational task that might be obviated with new data, a node could announce its intentions and pause for any proferred help. This could reduce processing needs greatly, especially in networks where sensor fields of view overlap significantly. A more generally applicable heuristic suggests delaying communication until currently available data have been integrated with existing known hypotheses. New data may produce new hypotheses or, more typically, reconfirm previously transmitted ones.

The answer ultimately depends on the relative benefits and costs of talking and thinking, and our experiments touch only a small area in this space. Further experimental work with specific network parameters and alternative heuristic strategies is needed to identify a good, general technique for DSNs.

IV. MACHINE STRUCTURES FOR DISTRIBUTED SITUATION ASSESSMENT

We now turn to the question of whether the preceding ideas can be implemented on a network of computer, rather than human, nodes. After all, human beings have a lengthy history of group problem-solving skills that are still poorly understood. Is it presumptuous to assume that we can transfer the heuristics of cooperation garnered from our experimental subjects to a purely machine-based organization?

Taking the best aspects of the two human structures as our guide, we compared the performance of both communicating and non-communicating network structures with a uniprocessor design employing the same heuristics. The design, implementation, and results of these experiments are traced below.

The difficulty of transferring human expertise to machine nodes made us look for simple and usable principles of cooperative behavior that could somehow guide the overall design process. One of the most important principles involved the use of models to simulate and predict other nodes' activities. A network design based on this principle—the processor assembly network (PAN)—is described at the end of this section. The PAN has not been implemented or tested, and we provide more questions than answers about its potential. Nonetheless, it appears to be a workable design for a network that manifests distributed intelligence.

EXPERIMENTS WITH MACHINE NETWORKS

We had several motivations for building a machine network: We wanted to see how difficult it would be to transfer the heuristics of human cooperation to a machine; we wanted a more controlled environment within which to compare structures; we needed a mechanism for developing more formal procedures for storing and sharing knowledge in a DSN task.

We continued to use the MPT as our domain, but because it was merely a symbolic stand-in for more general SA problems, the following discussion uses the more general terminology of sensing and tracking objects moving through space (platforms). We used the information hierarchy shown on the left-hand side of Fig. 6 in constructing data structures; the equivalent SA terms used in this report are given on the right. Note that the information hierarchy resembles other signal-interpretation decompositions [12].

Program Structure

Our basic approach was to construct a global problem-solver (which could see the whole puzzle at each round) using the cooperating experts paradigm [6]. This global program, extended with communication capabilities, formed the kernel of each node in a simulation of a sensor network. We simulated multiple processors, using the classic approach of time-slicing among processes, which requires no fur-

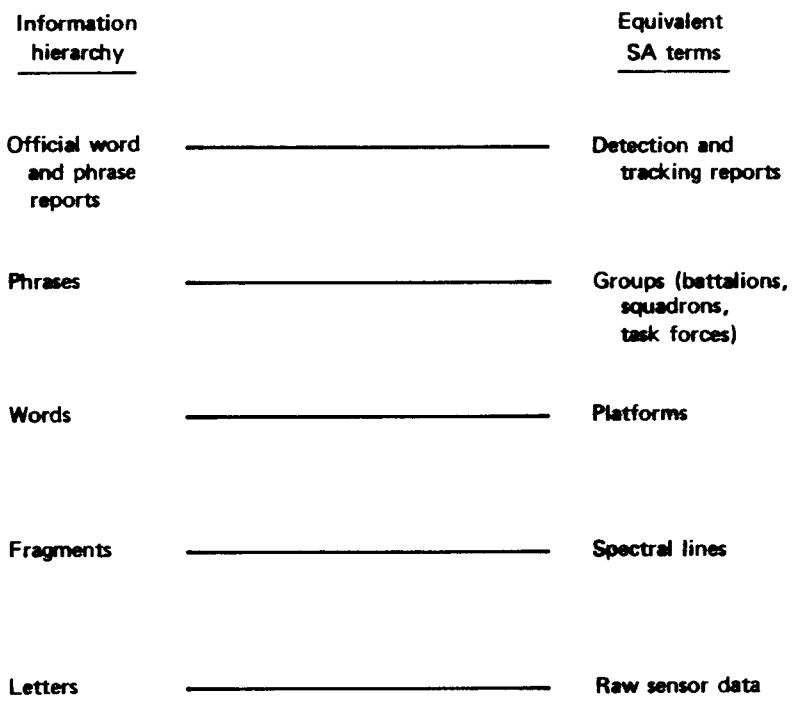

Fig. 6—Information hierarchy for model DSN

ther comment here. However, the technique of cooperating experts within a single process does require explanation.

As in the Hearsay-II [6] and HASP [12] systems, each node keeps a global data structure called a *blackboard* upon which information at different levels of abstraction is posted. This information consists of raw sensor input data and more abstract and less certain *hypotheses* about the specific spectral lines, platforms, and platform groups that might be deduced from those data. Information on the blackboard comes primarily from code modules called *knowledge sources* (KS) which reside at the node. Other nodes' information can, however, be posted on the blackboard by communication modules.

The KSs operate by taking as input hypotheses and data from the blackboard and posting new ones there. We can trace their operation by tracing the conversion of initial sensor data to a final report. At the close of each round, any data that have not been accounted for will be used by a KS specialist to spawn a series of possible spectral line combinations which it knows are plausible explanations for that new information. When this occurs, another KS responsible for converting spectral line hypotheses into platform hypotheses "wakes up" and posts its best guesses about platforms that might be generating those lines. As time goes by and new data come in, the confidences of these various hypotheses change, and one begins to stand out as the best explanation of the input stream. After a certain point, a hypothesis becomes good enough to share with other nodes in the system—a KS that "knows" about such criteria determines this and passes the hypothesis to the communication

module. Later, perhaps as confirming information from a neighbor arrives, the confidence in a hypothesis becomes so high that it is officially reported outside the network. When that happens, the initially competing hypotheses have been pruned away by a system of cooperating KS experts.

The communication restrictions imposed in the human experiments were relaxed in our machine-system design. Each node could communicate with any other node in the network, as well as with the outside world (via sensor data and official reports). The communication medium was perfect and instantaneous. Three types of information could be shared—low-level sensor reports, intermediate-level spectral line combination hypotheses, and high-level platform hypotheses.

Tests Performed and Results

Three basic structures were tested using this network simulation:

- *A global, centralized node.* This structure can also be viewed as a two-level hierarchy if we limit the operation of all lower-level nodes to the rote acquisition and retransmission of raw sensor data. This structure provided a benchmark against which to evaluate other network structures, since we thought that given enough processing power, a centralized algorithm would display the best possible problem-solving capabilities within a fixed set of KS modules.
- *A unilevel network of non-communicating nodes.* Simply reducing the field of view of the global node and creating a network simulation based on it allowed us to observe how performance was affected by strict decomposition with no communication or coordination.
- *A unilevel network of communicating nodes.* This structure—basically the AC—was expected to perform better than the network of mute nodes but worse than the global node.

We did not test a DHC-like structure, partly because of its poor performance during the human tests, and partly because computer resource and time limitations made it impossible to add more nodes and the considerably more complex "heuristics of management" which would have been required.

These limitations forced us to make simplifications even in what was implemented. The 15×15 square board used in the human experiments was reduced to 9×9. Each node thus had a 3×3 view instead of 6×6. The dictionary used was, of course, far smaller than that present in people's heads. We attempted to keep the discrimination problem intact in such a reduced dictionary by biasing each puzzle's dictionary with those words especially similar to the correct ones. For example, if "man" were correct, then the dictionary might include "may," "many," "me," "my," "human," and other words designed to propagate competing hypotheses for as long as possible.

Numerous different puzzles and degrees of noise injection were tried. Here, as in the human experiments, noise meant that some characters changed into their nearest neighbors—e.g., a *c* might be incorrectly sensed as a *b* or a *d*. A parameter determined the percentage of noisy incoming characters for each run. Generally, performance degraded gradually up to about a 50 percent noise level, at which point it dropped off severely. Figure 7 shows the results from a single puzzle in six

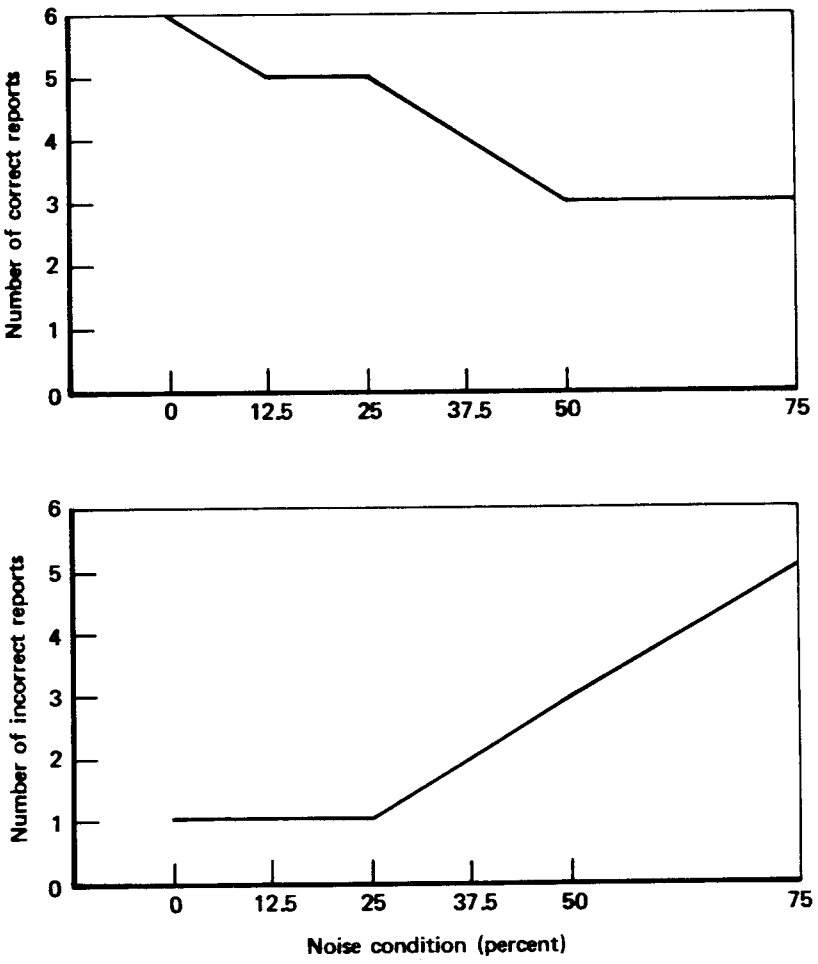

Fig. 7—Effect of noise on performance of machine global node

noise conditions ranging from 0 to 75 percent. At all noise levels, results were remarkably consistent across puzzles, enabling us to present the general results in Fig. 8.

The global node eventually produced human-like performance patterns (see (a) of Fig. 8).* It made reports at approximately the times when a human subject would have decided to do so; it reported prematurely (and wrongly) with a frequency again comparable to that of a human node; it remembered its guesses, so that no repeats were made.

As expected, the network structures generally performed worse than the global node in all categories except total solution time. Working alone, the global node required significantly more time to solve a problem—between 3 and 10 times more time than the slowest of the network nodes for each round. It also produced significantly fewer wrong and repeated reports than did either of the network structures.

*The heuristic nature of the program, coupled with the fuzziness of what "good performance" is, made it necessary to balance our reporting criteria. We chose human performance as our standard.

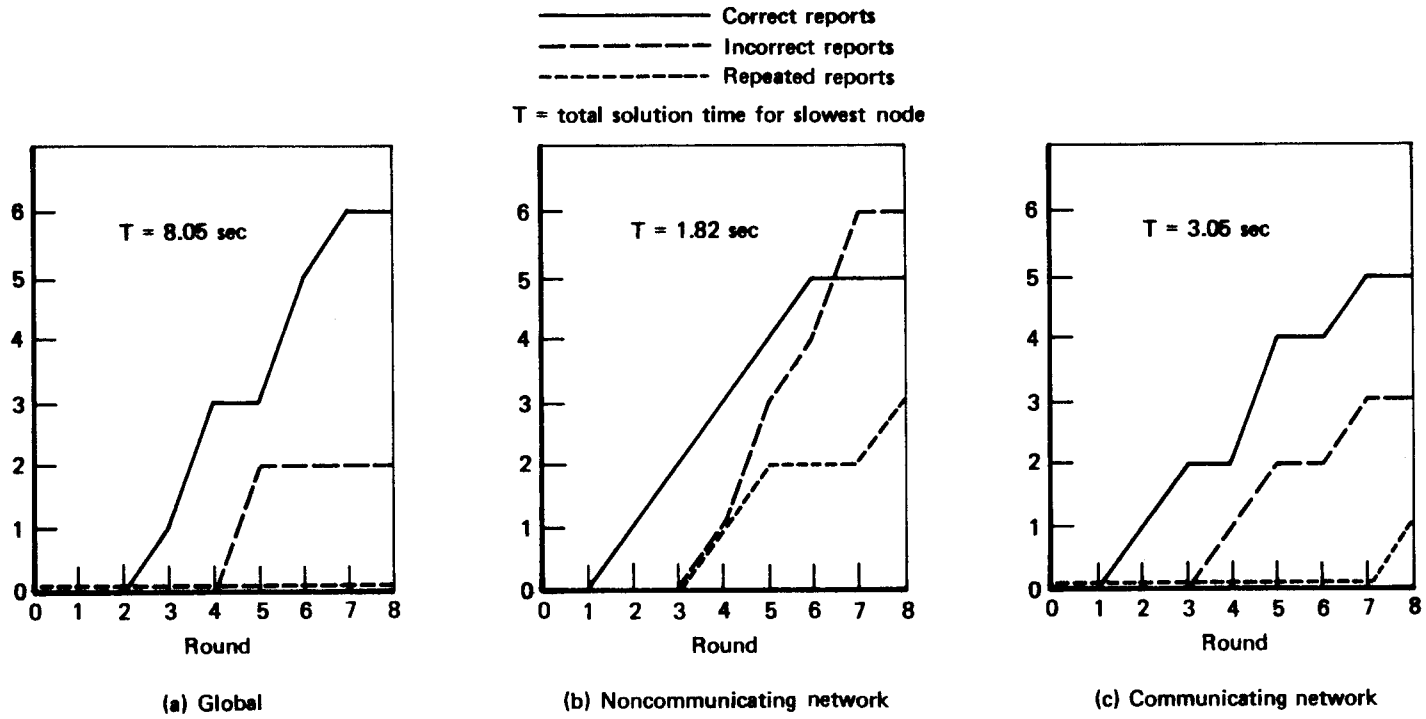

Fig. 8—Overall performance results of machine structures:
Problem 3, 25 percent noise level

However, graphs of correct reports are remarkably similar for all structures. The global node performed marginally better than the networks more often than not, but we were surprised to see a network structure outperform the global node even once. Perhaps the limited information available to each network node reduced the number of possible explanations at some points and thus actually improved reporting performance.

Another anomaly is the marginal difference that communication made among network nodes. Although sharing reported hypotheses did decrease the frequency of repeated reports, it did very little to improve overall performance. Yet communication costs a great deal—not only in execution time and space but also in the amount of time required to design and implement the required modules of code. This lack of difference in performance with and without communication can be partially explained by the fact that the first node in the path of a fragment entering from the outside was generally able to report it quickly, correctly, and autonomously. A domain with more interdependencies among the sensed elements would presumably require much more data and hypothesis-sharing than this one did.

A number of other deficiencies, both in the chosen simplified domain and in our implementation, should be noted here. For example, time-sliced simulation of parallelism is not a replacement for a real network of intercommunicating processors. We did some sensitivity testing by varying node activation sequences and found our results to be robust over such changes. But asynchronous nodes with imperfect, time-delayed communication links may produce very different behavior from that observed here.

Another deficiency stems from the simplistic design of our problem-solving algorithm. The MPT did not seem to require, nor did we attempt to create, any model-driven or "look-for" KS modules. As previously explained, KSs refine data in a bottom-up fashion: sensor data → hypotheses → reports. Higher-level hypotheses were never used to direct the acquisition of new lower-level data, to "look for" certain key confirming sequences. If processing and sensing resources were limited, as might be the case in a real-world DSN, such intelligent directing of the internal problem-solving process would become essential.

THE PROCESS ASSEMBLY NETWORK

Minimizing communication bandwidth requirements was a primary goal of our design efforts. In designing our machine network, we were able to greatly reduce message traffic by exploiting the machine's advantages. The machine nodes never had to ask for repeats or clarifications because they had perfect memory and communication. They never entered into dialogues in which a node requested data from another which then responded, because a protocol required nodes to share overlapping information with appropriate neighbors automatically. No general requests for assistance ("Anyone know a 5-letter word beginning with c-h-o?" or "Help! I'm overloaded!") were made because each node shared identical problem-solving knowledge and unlimited processing power.

The fact that our machine network achieved human levels of performance with such reduced communication loading prompted us to analyze the factors that motivate communication in a distributed intelligence network. Nodes talk to one another for various reasons:

1. To share raw local information that might have more global implications.
2. To update others' knowledge with newly arrived local information.
3. To direct the acquisition, processing, and transmission of another node's information, based upon local needs.
4. To subdivide and share the overall task load in a balanced way.
5. To share processing and reporting intentions to prevent redundant activities.
6. To share high-level inferences or hypotheses which may redirect or obviate the need for others' processing activities.
7. To predict where, what, and when new information will arrive to confirm or disconfirm existing hypotheses.

These reasons lead to two basic communication requirements:

1. Nodes must share time-varying information about the external world they are sensing.
2. Nodes must share time-varying information about their own internal states of processing.

This seemingly simple observation explains why communication requirements in our machine structure were so small. Implicit in the communication processing was the knowledge of what type of information would be forthcoming from other nodes, when it would be prompted, and how transmitted information would be used. In short, the nodes in our network implicitly shared models of each other's behavior.

This may not be possible, of course, with more complex systems performing more complex tasks. Moreover, the real world does not accommodate us by dividing itself into rounds that make update messages well-structured and convenient. It is continuously changing, presenting new aspects of its objects to us all the time, and knowing when something has changed sufficiently to tell another about it may be difficult indeed. If the world were steady-state, messages would not be required at all.

In that truism lies the key to a promising new concept in distributed problem-solving. The world is not static, so why should the hypotheses to explain it be static? What if hypotheses were somehow *active* themselves and could evolve over time in lock-step with the changing reality they purport to model? If the messages that nodes transmit contain both static data and dynamic procedures or models for changing that information over time, would not the need for frequent updating messages go away?

These questions have prompted us to design a generically different network structure, similar to Hebb's postulated organization of the human brain [13]. The design is based upon the concept of a "hypothesis process" as the basic system component.

To facilitate the explanation of how the PAN would work, let us assume a completely interconnected network of processors and sensors. When a new target is sensed, a hypothesis about it (classification, velocity vector, intentions, etc.) is formed and assigned to an available processor. This hypothesis is very different from those we have already seen, however, because it contains not only descriptive information about the target but also processing information about routine, expected changes it should undergo as the situation unfolds. As before, the formation of

this hypothesis process may prompt the creation of higher-level processes doing aggregation over space or more refined SA. Incoming information from the sensors will tend to confirm or discredit the hypothesis as a valid explanation for what is observed.

The advantage of the hypothesis process in this situation is that once information has been transmitted to each of these related entities—the lower-level information producers feeding it and the higher-level consumers using it—no further *routine* communication need take place. Each network element will contain models of the other elements that are relevant to its activities, and can thereby autonomously produce accurate expectations of their capabilities and needs. Only when non-routine, surprising information arrives should a message be sent to revise neighbors' models of what a node knows.

An example comparing our AC-like network with a PAN design will clarify this concept. Figure 9 shows a typical tracking task required of a distributed sensor network. Three sensors, S_1–S_3, have been deployed to protect three targets, T_1–T_3. Two groups of two attackers enter and form together into a V-wave which then proceeds as a single unit toward target T_3. We first describe how the AC-like network would react to this situation, then how a PAN would react.

In the traditional network, at time t, S_1 and S_2 would independently detect the incoming attackers and begin tracking. Each would notify its neighbors of the track initiation and the current location of the attackers—two messages to each of two nodes from the two sensing nodes, eight messages in all. At $t+1$, S_1 and S_2 would again notify their neighbors of the attackers they could see—a total of 16 messages. At $t+2$, another 16 messages would circulate. At $t+3$, S_3 would begin sharing its detection data and the total number of messages per unit time would rise to 24. In all, 64 messages would be exchanged in this simple example, each containing updated track information.

A PAN structure would not necessarily pair processors directly with sensors as above, but let us assume that it would, for simplicity's sake. The PAN begins at time t by creating hypothesis processes about the individual attackers detected and higher-level aggregation hypotheses about the two-attacker groups traveling together. S_1 and S_2 each transmit models of expected flight paths to their neighbors, enabling them to predict the targets' locations at any point in time. They will know when to expect the targets to penetrate their detection fields, as well as the targets' parameters of motion when that occurs. S_3 can immediately infer that S_1's targets may be headed toward target T_1 and that S_2's targets may be attacking T_2, and can begin directing defensive forces accordingly. A total of four complex messages containing the hypothesis-process models of aggregated target behavior have been transmitted.

At $t+1$, the targets are still traveling as expected and no messages are exchanged. By $t+2$, however, both S_1 and S_2 notice the formation of the V-wave and the new group direction of travel. Since each contains a model of the others' knowledge and processing state, a pre-established protocol to prevent redundancy selects S_1 to report this surprising development to S_3; S_2 knows S_1 will do so and keeps silent. Another message is thus transmitted. At $t+3$, the network's expectations are met as S_3 finally verifies what it has been told and marshalls defensive forces firmly around T_3. In all, five messages are exchanged.

This example illustrates the PAN's basic design principle: Additional computation (in the form of complicated hypothesis processes which must be constantly

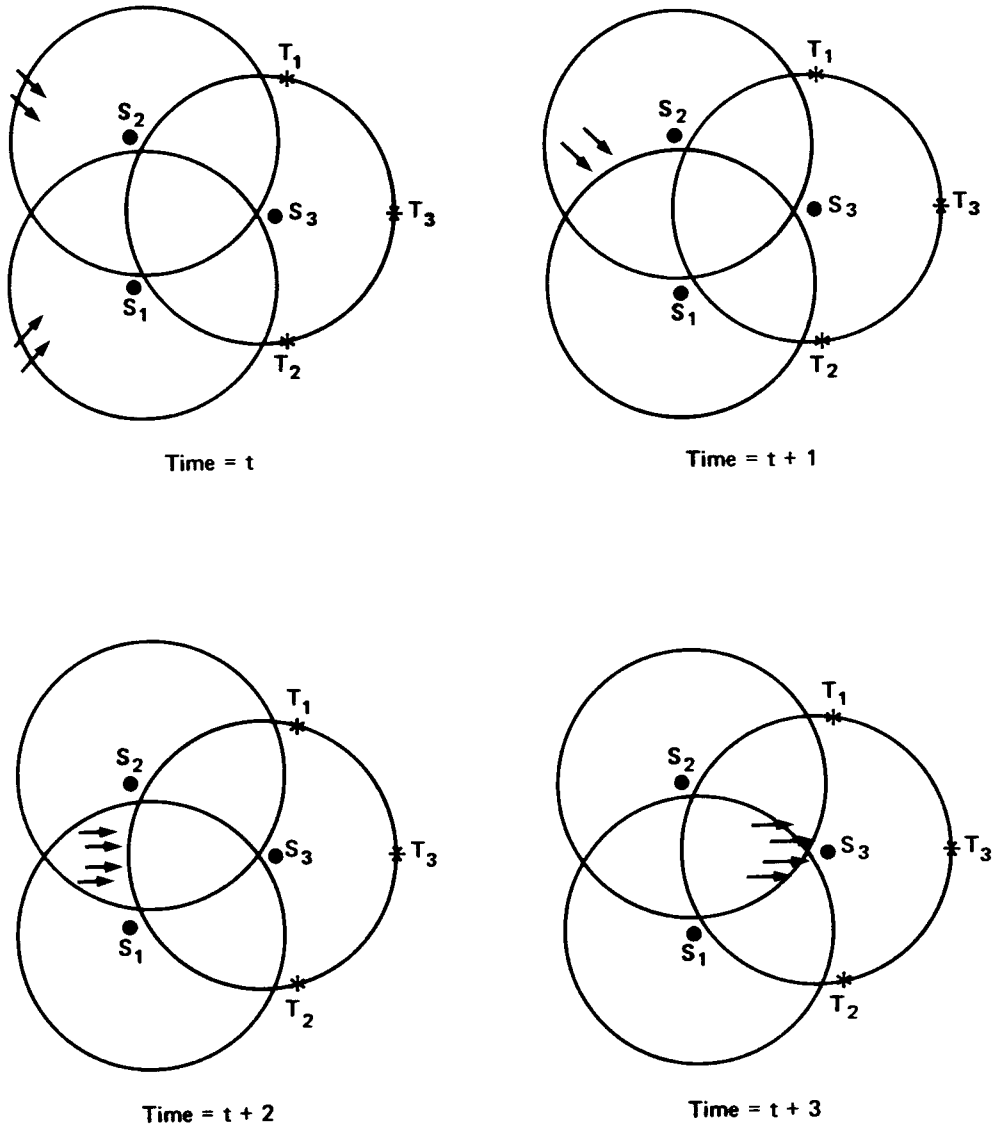

Fig. 9—DSN tracking task

maintained) can be used to reduce communication needs. The PAN relies upon two kinds of computation models that stem directly from our previous observations of information that nodes must share. Models of the sensed world and models of each other combine, enabling each PAN node to autonomously maintain predictions about upcoming events. These predictions reduce the necessity to share conforming information and leave the communication channels free for the transmission of surprising, unpredictable information. When messages of this type are used to revise and refine existing models, the PAN should evolve toward a steady-state operation with minimal communication requirements.

Naturally, the PAN design is preferable to the more traditional approach only if certain conditions are satisfied. Computational requirements are very great. Because each node must be able to model its associates, each must be a "generalist" with a full complement of problem-solving capabilities. This alone suggests that specialization may not be possible in a PAN, either in hardware or in software. Like any other distributed intelligence network, the PAN requires modules of domain expertise to transform incoming sensor signals into meaningful abstract reports. Unfortunately, these modules must also be communicable among nodes, since they form the basis for the hypothesis-process concept. We expect that a PAN implementation may employ coded references to common procedures with the hypothesis processes in order to reduce the sheer volume of material transmitted. Nonetheless, a "language" of hypothesis processes would be required in a PAN where none was overtly necessary before. Finally, the predictions that PAN relies on so heavily must be matched against sensor data with some regularity; knowing how often and how well this matching process must be performed is still a poorly understood problem of artificial intelligence.

The basic virtue of the PAN is its ability to perform distributed SA with minimal communication. In an environment where communications are limited, either by technology or by design (a quiet sensor net is less likely to be detected and jammed than a noisy one, for example), the PAN structure provides a promising design alternative. We have set forth only a superficial description of how it might function; major questions remain about its efficacy as an implementable, high-performance network structure for tasks of this nature.

DISTRIBUTED ARTIFICIAL INTELLIGENCE

The study of DSNs is merely the beginning of an entirely new field: distributed artificial intelligence (DAI). Where artificial intelligence to this point has studied and attempted to duplicate individual problem-solving methods, DAI transcends the limits of the individual and enters an entirely new dimension. Fields of study heretofore ignored by AI—organizational theory, sociology, and economics, to name a few—can contribute to the study of DAI. Presumably, DAI will advance these fields as well by providing a modeling technology suitable for precise specification and implementation of theories of organizational behavior.

Thus, DSN studies appear to herald a new and exciting direction in artificial intelligence. We expect cooperative problem-solving to become a central focus of future AI research, partly because of the trend toward system architectures that exploit multiple microprocessors, partly because distributed problem-solving is nat-

urally implemented as a system of cooperating experts, and partly because organizational theory is likely to benefit from computational approaches as much as cognitive psychological theory has benefited from computational attempts to model the individual mind.

Appendix

HEURISTICS FOR DISTRIBUTED SITUATION ASSESSMENT

This appendix lists some of the heuristics needed to perform the distributed SA task. These rules are expressed in general terms, rather than in terms of the experimental MPT domain from which they were derived. For instance, the objects under surveillance are referred to as "platforms," although they could in actuality be anything from personnel to cruise missiles. Refer to Fig. 6 for other equivalent terms used here.

In general, the model DSN of this appendix receives raw sensor data at various low-level nodes and transmits reports or declarations of platform motion out of the network. Nodes are arranged at various levels of authority and capability, with superior nodes generally integrating and abstracting the data they receive from lower-level nodes. Communication bandwidth is quite limited for the amount of information that might be shared.

Problem-solving in this hypothetical DSN proceeds neither exclusively top-down nor bottom-up. Rather, both procedures are represented in the heuristics below. Platform hypotheses, which contain both classification and tracking information, are created to account for multiple sensor reports. Recognizable spectral lines generate the platform hypotheses bottom-up, while existing hypotheses seek confirming or disconfirming information in a top-down fashion. Platform hypotheses near one another may suggest a group hypothesis. Either of these types of hypothesis may be reported or declared whenever confidence in it increases sufficiently.

Heuristics that embody this problem-solving technique appear below. Overall, they create or refine hypotheses for interpreting the data, or they direct the communication of information and tasks. The heuristics can be classified according to which tasks they support. The tasks include the incorporation of data into hypotheses, the identification of hypothesized objects, and the allocation and conservation of resources. To avoid a complex naming scheme, heuristics have been catalogued by their most important function. The following list shows the number of heuristics with specific functions, and the heuristics follow in the same order.

- Sensor data identification (1)
- Platform hypothesis
  - Formation (2)
  - Sensor data
    - Incorporation (5)
    - Scheduling (2)
  - Motion (4)
  - Identification (12)
  - Communication (1)
  - Deletion (1)

Groups of platforms
  Identification (1)
  Motion (1)
Sensor failure (3)
Communication
  General (4)
  Conservation (8)
  Scheduling (3)
Task allocation (9)

The heuristics are presented in the following format:

goal: what the heuristic attempts to accomplish
condition: characteristics of the world that must be true before the heuristic can be applied
action: what to do when the conditions are satisfied
reason: why this heuristic works and any beneficial side-effects

Sensor Data Identification

goal: identify spectral data
condition: the data come from the edge of a sensor's region
action: request identification at that location from other sensors
reason: sensor distortions, increasing with distance, may be resolved by pooling reports

Platform Hypothesis Formation

goal: create platform hypothesis
condition: processing of pre-existing platform hypotheses is complete and some spectral data remain unattributed to any extant hypothesis
action: create a new hypothesis for each local grouping of data
reason: such spectral data may have been produced by hitherto undetected platform(s)

goal: fractionate platform hypothesis
condition: the tracks of the spectral lines in a hypothesis form subgroups with separable tracks
action: form new hypotheses, one for each group, and delete the old composite hypothesis
reason: there seem to be multiple platforms, not one, as previously hypothesized

Platform Hypothesis Sensor Data Incorporation

goal: assign new spectral line data to an existing platform hypothesis
condition: there is an unassigned spectral line and there is an existing hypothesis "near" the signal source and that hypothesis contains that spectral line
action: assign the spectral line to the hypothesis and adjust the hypothesis position estimate to reflect the position of the new spectral line
reason: spectral lines follow platform motion

goal: assign new spectral line data to an existing platform hypothesis
condition: there is an unassigned spectral line and there is an existing hypothesis "near" the signal source and that hypothesis contains a spectral line very similar to the one observed
action: assign the spectral line to the hypothesis and adjust the expected spectral line slightly to reflect the new observation and adjust the hypothesis position estimate to reflect the new spectral line
reason: noise may change spectral line characteristics sightly

goal: assign new spectral line data to an existing platform hypothesis
condition: there are no unassigned spectral lines in the hypothesis' region to match predicted spectral lines and there is a nearby platform whose spectral lines may mask the predicted spectral lines
action: credit the hypothesis with the missing line
reason: spectral lines may mask each other

goal: assign spectral lines to platform hypotheses
condition: the region may be partitioned into disjoint clusters of spatially proximate sensor reports
action: consider each such subregion separately
reason: helps to control combinatorial search problems

goal: assign new spectral line data to an existing platform hypothesis
condition: the platform is entering the sensor's region
action: incorporate into the hypothesis spectral line data asising from the edge of the sensor field nearest the platform
reason: such data are likely to arise from the incoming platform(s)

Platform Hypothesis Sensor Data Scheduling

goal: schedule what to do next
condition: there are unassigned spectral data
action: assign new spectral line data to platform hypotheses known to the individual node prior to transmitting them or requesting related information
reason: it may obviate some communication needs

goal: account for spectral data
condition: always
action: assign spectral data to existing platform hypotheses before forming new hypotheses
reason: helps prevent formation of ephemeral spurious hypotheses ("ghosts") and makes more data available for testing existing hypotheses

Platform Hypothesis Motion

goal: predict position of platform hypothesis
condition: there is an established direction of motion for the hypothesis
action: predict spectral lines will move according to direction of motion
reason: platforms generally move in continuous trajectories

goal: predict position of platform hypothesis
condition: there is an obstacle nearby and the hypothesized platform is moving toward it
action: remove obstacle location from the set of possible positions and expect a change in the direction of motion
reason: platforms avoid obstacles

goal: identify constraints on platform motion
condition: the motion is near the boundaries of your sensor region
action: request obstacle location from adjacent nodes
reason: the motion of a platform near the boundary of a region can be influenced by obstacles just outside the region

goal: assist other nodes
condition: you have noticed global obstacle patterns
action: inform all nodes
reason: it may constrain their hypotheses

Platform Hypothesis Identification

goal: platform identification
condition: not all its spectral lines are identified
action: obtain the best partial matches from the set of platform signatures
reason: an informed guess is often better than no identification at all

goal: platform identification
condition: its spectral lines match a platform signature (S1) and its spectral lines are also a subset of some other platform signature (S2)
action: insure that the additional lines in S2 are not present in the data
reason: competing identifications must be isolated and disconfirmed

goal: platform identification
condition: not all its spectral lines are identified and a group type is known
action: obtain the best partial matches from the set of platform signatures in the group type
reason: the search for partial matches will be faster and the resulting set of possible identifications will be smaller

goal: platform identification
condition: the location of the hypothesized platform is also in the region of another sensor
action: request data on the platform hypothesis from the other sensor
reason: pooled data will lead to more accurate identification

goal: assign responsibility for platform identification
condition: information is needed from just one node
action: assign responsibility to that node
reason: reduces command redundancy and communication requirements

goal: declare an identification of a platform
condition: a platform has been identified and you do not have the authority to declare identifications

action: notify your superior of the declaration
reason: it will be passed to someone with the required authority

goal: make an identification of a platform
condition: a platform has been identified and the message system is very busy
action: declare the identification yourself, rather than sending it to a superior
reason: it is more important to declare identifications than to adhere strictly to communication protocols

goal: assist other sensors
condition: you intend to declare an identification
action: before doing so, broadcast the identity and track
reason: helps to prevent duplicate declarations by others

goal: assist other sensors and nodes
condition: a declaration is found to be incorrect
action: broadcast a correction immediately
reason: other sensors and nodes may be wasting resources and making errors as a result of believing the incorrect hypothesis

goal: assist other nodes
condition: you intend to cite a platform whose identity has been declared
action: also restate the node that identified the platform previously
reason: reminds nodes of platform's overall processing status

goal: unify global activity
condition: you are the superior node
action: require all declarations to be made by you
reason: most efficient way to keep whole system informed of declarations

goal: inform subordinates of platform declarations and conserve messages
condition: there are several identifications pending formal declaration
action: collect them into one message and broadcast it
reason: this information is not urgent and can be packed for efficiency

Platform Hypothesis Communication

goal: assist adjacent nodes
condition: a hypothesized platform is about to leave your region
action: send all information about it to the sensor whose region it is entering
reason: that sensor will best be able to monitor and track it

Platform Hypothesis Deletion

goal: disconfirm a platform hypothesis
condition: many of its predicted spectral lines are disconfirmed
action: delete the hypothesis
reason: lack of supporting data

Groups-of-Platforms Identification

goal: group-destination identification
condition: several platforms have been tracked

action: find a common pattern of motion and extrapolate it
reason: platforms are likely to travel in groups with a common purpose

Groups-of-Platforms Motion

goal: understand platform or group movement
condition: the processor is not busy
action: obtain positions of obstacles from other sensors and look for global patterns
reason: avoiding obstacles is a constraint on platform movement

Sensor Failure

goal: assist node with hardware problems
condition: a hardware problem has just arisen in some node
action: send the node a map of obstacles in and around its region
reason: its own processing capacity is better spent on reestablishing hypotheses than on such subsidiary tasks

goal: recover from a hardware error
condition: you have just recovered from a hardware error
action: broadcast your spectral line data immediately upon getting them
reason: you may go down again, but others may be able to use the data

goal: assist a node with hardware problems
condition: the node has just come back up
action: send the node copies of hypotheses in its region and such other useful information as global patterns
reason: help the node to reestablish its knowledge base

Communication

goal: obtain information
condition: you do not have the information and you suspect that another node does
action: send a request for that information to the other node
reason: you may get a reply containing the information

goal: compute a value
condition: you are busy
action: broadcast the problem, with a request for answers
reason: some idle node may solve it for you

goal: assist a node
condition: the node has made a request
action: respond in a timely manner
reason: data and results are perishable

goal: obtain a response from a node
condition: the request was sent a long time ago and an answer is still needed
action: repeat the request
reason: the node may have ignored the request or may not have received it

Communication Conservation

goal: conserve message capacity
condition: a message is not intended for only one node and broadcasting is possible
action: broadcast it
reason: saves processing associated with multiple message addressing and routing

goal: conserve processing capacity
condition: there is a message relating to distant events
action: ignore it
reason: those events will probably not affect you soon

goal: conserve message capacity
condition: there is a message to be sent and it is to be broadcast widely by your superior and you have more than one superior
action: send it to only one superior
reason: it will be broadcast with less redundancy

goal: conserve global processing and message capacity
condition: you are responding to a broadcast request
action: send a copy of your reply to all other original addressees
reason: obviate others' superfluous processing and messages

goal: conserve messages
condition: there are several messages to send and communication capacity is scarce
action: send to the union of the addressees a message that composes all the messages, preceding each with its specific adressee's name
reason: decoding packed messages is less costly than sending many independent messages

goal: conserve processing and message capacity
condition: you are very busy
action: do not attempt to integrate data that other sensors can also process and do not send detailed status reports to your superior
reason: these are less important parts of your task

goal: conserve messages
condition: you are preparing a message for a node who is known to be very busy
action: delete the message
reason: the busy node will not have time to process it

goal: conserve processing and message capacity
condition: system organization is hierarchical
action: do not reconfigure authority structures dynamically
reason: such reorganization cuts into subordinates' capacity to handle essential task activities

Communication Scheduling

goal: schedule
condition: there are many unprocessed messages
action: process them
reason: other processing would probably benefit

goal: schedule
condition: there are few unprocessed messages and there are data to process
action: process data
reason: improving hypotheses is probably more valuable

goal: process a message
condition: always
action: respond as quickly as possible
reason: a late answer may be valueless, in which case the effort expended on it would be wasted

Task Allocation

goal: assist in allocation of responsibility
condition: action has just commenced and the processor is not busy and there has been no message from your superior asking that raw data not be sent
action: send the superior the entire set of spectral lines and positions
reason: the superior may have sufficient resources to develop a detailed understanding of its large region

goal: assist in allocation of responsibility
condition: action has just commenced and the processor is busy or there has been a message from your superior asking that raw data not be sent
action: send the superior the number of data reports and identify all of your busy borders
reason: the superior may be able to derive a general understanding of its region even from such abstractions

goal: assist in allocating responsibility
condition: you are short of messages or processing capacity
action: inform your superior
reason: your superior may be able to lighten your load

goal: allocate responsibility
condition: action has just commenced
action: assign some subordinates to two or more superiors
reason: provides redundancy of command and supports communication over borders between adjacent command regions

goal: allocate responsibility
condition: always
action: request business status from subordinates and reallocate so that business is evenly distributed
reason: greater overall throughput and reduced bottlenecks

goal: allocate responsibility
condition: a sensor's region is inactive
action: deallocate all responsibility for this region
reason: nothing is going on there

goal: assist in reallocation
condition: you are idle
action: report to superior
reason: allows him to understand workloads

goal: assist in reallocation
condition: you are busy
action: report to superior
reason: allows him to understand workloads

goal: assist a very busy subordinate
condition: you have an idle subordinate
action: send the busy node's data to the idle node for processing and tell the idle node what to do with them
reason: share tasks over available resources

REFERENCES

1. Lacoss, R., and R. Walton, "Strawman Design for a DSN to Detect and Track Low-Flying Aircraft," *Proceedings of the Distributed Sensor Nets Conference*, Carnegie-Mellon University, Pittsburgh, December 1978, pp. 41-52.
2. Smith, R., and R. Davis, "Distributed Problem Solving: The Contract Net Approach," *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, July 1978, pp. 278-287.
3. Sanderson, L., and J. Lord, "Microcomputers Promise Less Stop, More Go," *IEEE Spectrum*, November 1978, pp. 30-35.
4. Kahne, S., I. Lefkowitz, and C. Rose, "Automatic Control by Distributed Intelligence," *Scientific American*, Vol. 240, No. 6, June 1979, pp. 78-90.
5. Hayes-Roth, F., and R. Wesson, *Distributed Intelligence for Situation Assessment*, The Rand Corporation, N-1447-ARPA, January 1980.
6. Erman, Lee D., F. Hayes-Roth, V. R. Lesser, and D. Raj Reddy, *The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty*, Carnegie-Mellon University Technical Report, CMU-CS-79-156, Pittsburgh (in press).
7. Lawrence, P. R., and J. W. Lorsch, *Organization and Environment*, Harvard University, Cambridge, Massachusetts, 1967.
8. Uhr, Leonard, *Pattern Recognition, Learning and Thought*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
9. Applewhite, C. M., "Distributed Computer Architecture for the Discrete Address Beacon System," *Proceedings of the First International Conference on Distributed Computer Systems*, Huntsville, Alabama, October 1-5, 1979, pp. 480-489.
10. Hayes-Roth, Frederick, "The Role of Partial and Best Matches in Knowledge Systems," *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (eds.), Academic Press, Inc., New York, 1978, pp. 557-574.
11. Kahn, R. E., "The Organization of Computer Resources into a Packet Radio Network," *IEEE Transactions on Communications*, COM-25, No. 1, January 1977, pp. 169-178.
12. Nii, H. Penny, and Edward A. Feigenbaum, "Rule-Based Understanding of Signals," in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (eds.), Academic Press, Inc., New York, 1978, pp. 483-501.
13. Hebb, D. O., *The Organization of Behavior*, John Wiley & Sons, Inc., New York, 1979.

RAND/R-2560-ARPA