Data Communications

Feature AI carves inroads: Network design, testing, and management --- A survey of artificial intelligence work in progress reveals a medley of prototypes (and a few products) plus a hint of what's to come. --- Lee Mantelman, DATA COMMUNICATIONS

7817 words
1 July 1986
Data Communications
DCOM
Pg. 106
Vol. 15, No. 8
English

Yes, it's in another headline. Artificial intelligence (AI) has made the leap from an obscure, ivory-tower chimera to an industry hot button. With the press touting its wonders and vendors claiming to have built it into their latest products, many users are understandably befuddled. To be sure, AI does have an important place in networking. By looking at the projects under way at a number of organizations, users can more critically appraise what role AI will play in their own networks.

No doubt the AI industry is growing fast, which explains the flurry of press coverage. Specialized vendors of AI processors, such as Symbolics Inc. (Cambridge, Mass.) and Lisp Machines Inc. (Los Angeles, Calif.), are beefing up their high-end development products and coming out with lower-priced entry-level models. Mainstream computer and communications vendors including Digital Equipment Corp. (DEC), Texas Instruments Inc., Xerox Corp., and Tektronix Inc. are heavily involved, producing AI workstations and striving to integrate the new technologies into their existing product lines. From the low end, microcomputer versions of AI languages are practically falling off the shelves, giving users much easier access to the ideas and potential of artificial intelligence.

Why AI (and just what is it, anyway)? Using AI to solve networking problems, however, is still a new and fairly sparse area. Only a handful of companies have moved beyond theory to produce prototypes or purchasable products. Most of these

outfits are large, well-financed research organizations. Remember, though, that these are merely the first small steps. As the tools and concepts of artificial intelligence spread in ever-widening circles, users of information networks are going to be faced with giant leaps of opportunity and challenge.

While there is no universal agreement on the definition of the term, AI refers, at least in part, to a set of programming languages, techniques, and tools that allow computers to exhibit what might be considered intelligent behavior:

--Just as high-level languages (such as Fortran, Pascal, and Basic) allow programmers to couch machine instructions in semantically meaningful commands, AI languages (including Lisp, Prolog, and OPS5) permit information at the level of facts, beliefs, and knowledge to be translated into computer commands and stored in "knowledge bases."

--Such techniques as intelligent user interfaces and object-oriented environments allow users to interact with and benefit from this information.

Other techniques, such as inference engines in Prolog, turn rule bases from passive information, as would be stored in a book, into interactive sources of relevant help about given problems. They may do this by scanning search trees consisting of rules.

--Several AI tools are gaining momentum outside academia, mostly for constructing expert systems (programs that function in a certain field of knowledge, or domain, as a human expert would). An established line of development tools, such as the Automatic Reasoning Tool (ART) from Inference Corp. (Los Angeles, Calif.), the Knowledge Engineering Environment (KEE) from Intellicorp (Palo Alto, Calif.), and S.1 from Teknowledge (also in Palo Alto), is helping to bring expert systems into more widespread commercial use.

Of all the topics studied by AI theorists, expert systems are the most likely to make a big splash in the networking world in the near future. A tremendous amount of knowledge is needed to design, operate, and troubleshoot networks. Bruce Johnson, partner-in-charge of AI practices worldwide for the consulting firm Arthur Andersen & Co., sees business conditions as ripe for a knowledge-based approach to network management. He points out that telephone companies and other large corporations that operate their own networks face problems in common:

--The networks are becoming unique, complex, and heterogeneous, with no overall network control.

--Skilled people to manage these networks are scarce and may change jobs frequently, making programs that can capture their knowledge attractive.

However, the fact that much of the needed information about networks is available in electronic form suggests that systematizing network operations is feasible.

As vendors incorporate AI into their products and services, says Johnson, network managers are likely to benefit from improved network reliability (including service, line quality, and up-time). A potential for cost savings might also exist. For example, the ability to diagnose problems and get lines back up faster might reduce the need for redundancy in the network.

Networking products that use AI are already starting to show up, and Johnson expects more to be available within the next year or so. Applications built with AI technology are likely to be more flexible and expandable than others, and these qualities can add a competitive edge to users' networks. Does this mean that users should learn Lisp? Not necessarily. Johnson points out that "only the very largest organizations will be concerned with building these applications; everyone else will just buy them. AI will probably be an invisible part of these applications."

Designs on your network Companies that are serious about developing expert systems to help them manage their networks will probably need a blend of expertise, at least if they want to use the tools and languages currently in favor. In addition to data communications savvy, a firm might need the resources of an in-house research organization or, barring that, a relationship with one of the leading AI schools (such as MIT or Stanford).

Bolt Beranek and Newman (BB&N) in Cambridge, Mass., is blessed with all of the above. BBN Communications Corp. develops and markets private network products. Also, in addition to BBN Laboratories, the company's research arm, BB&N has close, longstanding ties to MIT. By putting the right heads together, the firm has developed what it refers to as an "expert assistant" tool, called Designet, which helps design data networks. BB&N workers are currently using this tool for customer projects.

Designet aims to supplement the limited supply of skilled network designers. It allows an operator who does not have a Ph.D. in mathematics to develop network-design solutions that approach the optimum. The practical upshot is that designers spend less time overcoming the limitations of their tools or doing manual drudge work, such as tracking facts about the design in progress. With the intelligent design tool, they

can determine the basic topology faster. They can then spend more time doing iterations, looking at alternatives, and honing the design.

Artificial intelligence programming was used to construct the network model that supports Designet, while good, natural common sense was brought to bear on the human interface. Much thought was given to how the tool should present information. Designet's developers observed different network designers using different kinds of strategies. Some liked to manipulate pictures, others preferred working with tabular reports. Almost all would sit down in front of a table or a desktop covered with papers.

The developers wanted to provide multiple ways of displaying network information so that users could call up a tabular, geographic, or logical network view. Designet, which runs on a Symbolics Inc. 36XX (an AI workstation with over four megabytes of memory), uses two screens, one black-and-white and one color. It exploits windows, color graphics, and a flexible menu scheme.

Step by step The main console consists of several windows, or "panes" (Fig. 1a). Utilization reports are displayed in a report pane (top left). Below that is a narrow summary pane showing the state of the network. The lower half of the screen is dominated by a large Lisp interaction window for textual reports and code. The darkened line at the bottom is a mouse line, detailing what will happen if the left, right, or center mouse button is pressed. In the upper-right-hand corner, a command window changes based on the design's current phase. Below that are menus for generating reports and manipulating the color map. To the left of those is a "mouse hole" for switching between screens. When the user moves the pointer into this pane, the pane darkens as shown, the black-and-white console is suspended, and the pointer appears on the color console.

A network design using Designet takes place in four phases, the last three of which can be represented on the color monitor:

--The customer provides data describing network requirements. This data currently comes from customer surveys. Getting this type of information is difficult. Customers do not know what kind of users or traffic will occur on their as-yet-nonexistent networks, making the design process largely a work of estimation. Besides, with blatant disregard for the best-laid plans of designers with mice, people invariably find unexpected ways to use a network.

--After an approximation of basic network needs emerges, the first problem is the clustering of terminals, attaching them to packet assembler/disassemblers (PADs) or terminal-concentrating equipment. Since they are relatively cheap, many PADs are often used to serve small circles of terminals, eliminating the need to run longer lines (Fig. 1b).

--In another clustering problem, the PADs and the hosts must be linked to packet switches (Fig. 1c). To choose node locations, the packet switches may be added to the color screen manually.

--Finally, the packet switches themselves must be connected to a backbone network.

The sequence in Figures 1 and 2 shows part of a typical backbone design. Since there is no such thing as a perfect solution to the intractable network-design problem, Designet uses heuristics, or rules of thumb. For example, when people design networks, they usually start with a minimum spanning tree to ensure that all points in the network are connected. Designet does this automatically when the user gives the command "make connected" (Fig. 1d). A biconnected graph, which provides two paths to each node, can also be generated (Fig. 1e) and displayed. Figure 1f shows multiple views of the design so far. The biconnected graph is in the upper-left-hand corner. Note that selected portions of the map can be enlarged, such as the Great Lakes area shown at the lower right.

To test a design, the operator runs a mathematical simulation of the current configuration. The simulation uses a model of the routing algorithm and assigns flows to paths that will cause minimum packet delay. In this way, it can compute loading and, hence, performance. Results are displayed above the map, next to the blue, red, and green scales showing, respectively, the maximum line and packet-switch utilizations and the total cost of the configuration. (Note that communications lines are blue on the map.)

After the operator adds a few lines that intuitively seem beneficial, the simulation results show that the average line utilization, which the operator would like to minimize, has gone down substantially while the cost has increased only slightly (Fig. 2a). As shown in the figure, the network can also be displayed logically. Stripping away the geography from the topology can be useful in backbone designs that are more complex than the example given here.

As a quick way to enhance the design by improving network performance, an algorithm can be run (Fig. 2b) that reduces the diameter of the network (a measure

relating to hops between nodes). Running the simulation again shows another small cost increase and a noticeable, though less drastic, performance gain (top of Fig. 2e).

To further refine the network, the operator may try to delete costly lines that do little to improve performance. Selecting a line utilization report (Fig. 2c), the operator chooses a line from New Orleans to Atlanta as a "cutpoint" (Fig. 2d). Lines are divided into two groups, those utilized more than and less than the cutpoint. Cutpoints illustrate the interplay of graphic and tabular information. Since the table is sorted, all lines above the cutpoint on the list are busier than the cutpoint is. Also, on the color map, all less-utilized lines are dimmed while those with more traffic are highlighted.

The user decides to delete the long line from Texas to Maine, since it is below the cutpoint. Selecting the line on the color display (Fig. 2e) and deleting it (Fig. 2f) brings down the cost. Now, however, the network is no longer biconnected, as the Texas node could be isolated due to a single line failure. In response, the operator manually adds a less-expensive line from New Orleans to Texas.

Objects and projects As illustrated, whether network entities are shown as names in a text display, items in a table, or icons on a screen, all representations can be manipulated similarly. This is accomplished by storing information about multiplexers, switching nodes, protocols, tariffs, network algorithms, and so on as objects in Zeta-Lisp (an object-oriented programming language).

One future goal for Designet is the addition of an easier interface for updating the knowledge base, so that nonprogrammers can define new types of objects. Another is to move the design and analysis process into network management and operations. As demand services, such as switched 56-kbit/s trunking, become available, network design becomes a real-time problem.

Designet will be part of an integrated line of intelligent tools. A knowledge-acquisition expert system called Specifier, now in prototype stage, will replace the customer surveys in the first phase of the network design. Users will thus be able to describe their network requirements directly to Designet.

Designet's output will be given to Configurer, an expert system being developed to configure node sites (including placement and parts lists for boards, racks, cables, and so on). Configurer is similar to R1, an early expert system developed by Digital Equipment Corp. to configure VAX minicomputers.

To market, to market The knowledge within expert systems comes into play as the programs guide or assist users in accomplishing a task, such as designing or configuring networks or nodes. Network diagnostics and testing also lend themselves to this kind of electronic consulting.

Avant-Garde Computing Inc., a network management company, collaborated with several consultants from MIT on its Net/Adviser (NA) expert system. According to the company, the first installations of NA will occur this summer at the Metropolitan Life Insurance Co., the Prudential Insurance Company of America, Shearson/American Express, and Ford Motor Co. The expert system is designed as an optional enhancement to a conventional product, Net/Command (NC), which was developed in parallel with the expert system.

Net/Command provides central consoles, or workstations (Fig. 3), from which an operator can access various network monitoring and control tools (including IBM's Network Communications Control Facility and Omegamon programs, such modem monitors as Codex Corp.'s Distributed Network Control System, matrix-switch control consoles, and so on). Each tool provides an alert mechanism (a log or a display of errors), which NC receives and enters into a centralized database of alert information.

Then Net/Command prioritizes and filters out noncritical, no-action-needed alarms (using conventional programming techniques). Filtering can be done based on various kinds of levels: level of immediacy (problems that require immediate action, such as a line that has stopped polling), level of device (for example, only problems related to devices higher than, say, a control unit in the network hierarchy), or level of occurrence (thresholds). NC places alerts that need special attention at the top of a list on the appropriate operator's terminal.

The Net/Command package consists of one or more Sperry or IBM microcomputer-based workstations attached to proprietary hardware called an access unit (AU). The AU has multiple microprocessors that each handle a variety of ports. NC, which is now in testing at a beta site, costs from $20,000 to several hundred thousand dollars, depending on the number of workstations and network control tools.

Software components The expert system, Net/Adviser, starts at around $200,000. Two parallel processes run within it. One, a Lisp process, is constantly receiving information from the access unit in the form of alerts and posting them to its network configuration database. The Lisp database, like the one in Designet, maintains a

comprehensive understanding of the components and how they relate to each other in the network configuration.

Each component is maintained in the database as an object. The Lisp code changes the properties (called "instance variables") of an object when the corresponding component's status changes. Lisp creates the relations between the objects and, hence, determines the network configuration by interpreting information supplied via the various tools (such as IBM's Network Control Program generation or "NCP gen").

The other process, an inference engine written in Prolog, operates on a database of rules of the form, "make the following recommendation if the following status exists or alert occurs." The Prolog process continuously reviews its rule base to see if it can offer a relevant piece of advice. A custom interface to marry the two languages was written by an outside consulting firm.

The access unit is constantly communicating with the expert system. The unit sends alert information to the expert system and then receives certain recommendations back from it.

After selecting a problem to work on, an operator can ask NA for advice. Rather than comment on all the different problems in the network, the expert system provides only advice that is germane to the particular alert on which the operator is working. It can also offer all related alert messages, including information from the tools that monitor the host, the modems, and so on.

Optional: Roll up your sleeves Net/Adviser was developed on a Symbolics 3640 AI computer, although it is delivered to the user on a smaller model, the 3610. The user interface to NA is not a Symbolics terminal but the same network workstation used in Net/Command. After startup, no direct interaction with the Symbolics machine is required. However, Avant-Garde believes that some users, though by no means all, will want to use its product as an entry into AI-for-networks development work. Users who want to maintain or work with the AI portion will need the more costly 3640, through which the knowledge base and the inference engine are accessible. No price has been set for this package. Again, the user would probably need an AI department (or at least someone who understands Lisp and Prolog). More timid (possibly more sensible) users or those without sufficient resources can get the less flexible but more economical Symbolics 3610.

While Net/Adviser currently gives only advice, Avant-Garde says it may have the expert system become active in the network as a second phase. Then, instead of

simply giving a recommendation to start up a node or switch a line, the expert system will pass such a command to the appropriate device.

Maintenance mavens A group of researchers at Bell Communications Research (Bellcore) is taking a tack similar to that of Avant-Garde but splitting the functionality into several expert systems. For example, knowledge of what to do about problems, as in Net/Adviser's rule base, is embedded in Bellcore's consultation expert system, Smart. Also, like Avant-Garde, Bellcore is planning to actively involve the stored knowledge in the network using a package called React.

Whereas BB&N produced a tool for internal use and Avant-Garde developed a product for sale, the Bellcore group's only output is a set of specifications or requirements for expert system applications in network maintenance. This may include development of a prototype, as in the case of Smart, the Switching Maintenance Analysis and Repair Tool, an expert system that assists in diagnosing 1AESS switches.

In the distant past, a person would stand in front of a network element, such as the 1AESS, and manually perform such operations as detecting trouble and restoring service. Eventually, a minicomputer-based Operations Support System (OSS) was developed and deployed in the network. The OSS automatically analyzed data from a number of network elements (Fig. 4). Remote interfaces on the OSSs allowed one switching control center (SCC) to handle a large number of switches.

If they could isolate a particular problem, people on the SCC staff would either fix it (if it was software-based) or dispatch a repair crew (if the hardware was at fault). If the trouble spot could not be located, the operations people would escalate the problem by enlisting the help of expert switch consultants.

The Smart expert system intercepts and offloads problems for these consultants. To acquire the knowledge base for Smart, Bellcore worked with experts from Nynex, Bell Atlantic, and Southwestern Bell. Pooling the knowledge of several regions gave Smart a broader base. For example, the Nynex workers had seen a lot of metropolitan area network-style problems, whereas the Southwestern Bell people had had more experience with weather-battered lines.

Getting ready for the big time Bellcore initially used Tecknowledge's S.1 Lisp tool for Smart. Several months ago, Tecknowledge announced that it was porting S.1 to the C language. Bellcore has already translated the bulk of its code (all the software that

acts as the liaison between the SCC, the OSS, and Smart) to C under Unix with a view toward field deployment.

Smart currently exists as a demonstrator running on various machines. Bellcore is developing the demonstrator into a prototype that will be installed in Bell Atlantic's Baltimore operations center by the fourth quarter of 1986. However, Bellcore plans to make the demonstrator available to the three "friendly users" that helped create it (Nynex, Bell Atlantic, and Southwestern Bell) to get feedback before building the prototype.

With the demonstrator, people answer the expert system's questions using information they obtain from the OSS. However, Bellcore intends that the prototype, when developed, will get its information directly from the OSS. To do so, it will have to simulate a user of the OSS by speaking OSS's command language.

Although Smart works with 1AESS switches, the Bellcore group believes that the same approach can be used to develop knowledge bases to support other network elements. Bellcore will attempt to construct prototypes for these elements.

More help Smart takes some of the load off the expert consultants by fielding questions from the analysis people. Another expert system, React (the Real-time Expert Analysis and Control Tool), is intended to serve as a front-end to the network control personnel.

In the SCC, controllers watch a big board for alarms and decide which are problems. They turn the problems over to the analyzers, who fix them if they can and otherwise call the consultants. The ultimate objective of React, which will be phased in over time, is to monitor alarms coming in from the network, determine which alarms are worth responding to, pass them on to Smart, write up trouble tickets, and suggest corrective action.

React is in an earlier stage than Smart. Presently, React is a demonstrator running on a Symbolics 3640 written in Intellicorp's KEE. It gets its incoming alarms not from a real source but from a simulation, a series of random selections from a canned alarm file. The demonstrator as yet has no rules built into it but, like Designet, makes sophisticated use of objects to classify components in the real world and their relationships and interactions.

Bellcore says it has some fairly grand plans for React but will take them one step at a time. As with Smart, React will focus first on the 1AESS. React ignores alarms that have occurred before or that are known to be under repair or out of service and

processes the rest according to set procedures, recommending corrective measures. Eventually, React itself may actually carry out these repair actions.

Like Eve, like Adam AT&T Bell Laboratories, from whose rib Bellcore was created, is also developing expert systems for managing networks. The Network Management Expert System (Nemesys) is designed to fight congestion in AT&T Communications' long-distance public network. Nemesys is based on techniques that optimize network performance during periods of overload and stress.

Historically, real-time network optimization was done by operators who said "please call later" if circuits became too busy. Later, a Network Operations Center (NOC) was built so that operators could reroute traffic by sending commands to the switches. Today, a tremendous amount of constantly changing information is presented to operators on a wall display and on monitor screens at the NOC in Bedminster, N. J.

The complexity of this environment is increasing because the network topology is changing from a tree-shaped hierarchical routing structure to a nonhierarchical mesh network. Eventually, nonhierarchical routing will embrace all of AT&T's large 4ESS switches. Also, as new services are added to the network, loading conditions will change more dramatically over shorter periods of time.

This environment presents new challenges. Only a few experts are capable of engineering for these more complex networks and their time is constrained. AT&T Communications would also like to capture their expertise in case their number is reduced. It is, therefore, sponsoring a Bell Labs effort to develop an expert system that will help operators manage the network in real-time.

At present, status information about how many calls were completed or blocked, and why, is gathered from the switches and presented to the NOC via the Network Management Operations Support System (Nemos), a distributed collection of AT&T 3B20 minicomputers. The Bell Labs group has built what it calls an "evolving prototype" of an expert system to make suggestions and eventually to take action.

Pattern recognition problem Usually, operators will try to reroute calls around overloaded lines. The expert system, which runs on a Symbolics 3670 Lisp machine, tries to recognize what are known as focused overloads, where many callers are trying to reach a single location. For example, if there is a tornado in Atlanta, thousands of people call friends or relatives who live there. In such a case, it does not make sense to seek new ways to route traffic to Atlanta, because either there are not enough switches serving Atlanta or the local network in the area is overloaded.

The expert system will then step in to tell the network or the operators not to fix the problem (because nothing they can do will cure a focused overload). It will also determine which lines are candidates for rerouting, which have highest rerouting priority, and which spare lines are available.

Although Nemesys is designed for field use, several factors are impeding its deployment: Will it solve problems fast enough? Will the NOC organization trust the expert system to assist them or to do part of their job? The most serious concern is how to get Nemos and Nemesys on speaking terms. Bell Labs admits that it is not yet sure how to do that.

Meanwhile, another group at Bell Labs is working on a different expert system to find and troubleshoot faults in large, geographically dispersed Datakit networks, which are Centrex-style LANs. This project is practical for Bell Labs, which uses Datakit to connect 11 locations in New Jersey, three in Illinois, one in Ohio, and one in Pennsylvania. In early 1986, the network had 99 nodes and 1,300 ports. The Datakit network-management expert system, Troubleshooter, is unique in that it learns over time.

Just as Net/Adviser needs Net/Command to tie into a real network, a centralized network management tool called Starkeeper forms the basis of the Troubleshooter expert system. Starkeeper collects performance, trunk, and billing data for the entire Datakit network. As alarms occur, a color graphics display changes dynamically. The expert system connects to the network through Starkeeper, which it uses for examining the status of the network.

The goal of Troubleshooter is to allow a novice network administrator to function as an expert diagnostician. To use the expert system, the administrator answers questions about a problem. Troubleshooter runs through the statistical likelihoods of various causes of the problem and tells the administrator what it considers to be the probable cause (after first running diagnostics to be certain of its assessment).

A hierarchy of likelihoods Troubleshooter operates according to conditional probabilities. When a symptom is described, Troubleshooter formulates a hypothesis about the fault location using both heuristics and empirical data. The conditional probabilities change dynamically. Since its search tree is based on probabilities, Troubleshooter's search strategy will adapt and change as facts become known. To be efficient, the expert system begins its search from the most probable fault. If it does not find a fault at a higher level, it starts looking at less likely possibilities.

At each step, Troubleshooter decides what tests to do, does them, and, if it can suggest a solution, reports the results back to the administrator. If the administrator acts and corrects the problem, the expert system adds that fact to its trouble-history database. The expert system learns by means of this database, which is initially seeded with the probabilities that given faults are in given locations. These probabilities are determined based on the experience gained by customer service personnel in answering calls from users on an "800" number.

As problems are fixed and as the network grows, the log is updated and the probabilities change. If, for example, there is an abundance of problems on a particular type of board, the probabilities will change such that the expert system will identify that board as the most probable cause of new problems.

Troubleshooter is a prototype. It is presently being deployed at Bell Labs' sites with large Datakit networks in order to tune the human interface. The software was developed on a 3B2 minicomputer in Franz Lisp, which runs under Unix. It was later ported to a PC7300 microcomputer. There are plans afoot to make it a product. Its primary market would be such users of Datakit as telephone operating company central offices and certain users of large internal networks (the government, for instance).

Contemplating their cables A third Bell Labs expert system for networking, probably the oldest and best-known of all such programs, is ACE (Automated Cable Expertise). It is also the only program besides Net/Adviser that is available as a product. Marketed to telephone companies, ACE acts as an intelligent assistant to a cable analyzer.

Telephone companies run cable analysis and rehabilitation to save money and improve customer satisfaction. When poor workmanship, natural damage, or deterioration from rain, squirrels, or rodents plague a line, repair technicians often make short-term fixes that solve the symptoms but not the underlying problems. Since such problems can build up over time and form recognizable patterns, telephone companies analyze repair histories to try to identify trouble areas.

To do this, a cable analyzer would pull data from the Cable Repair and Administrative System (CRAS). However, learning to find problems from this historical database can often take five to seven years of training. Even then, the analysis is not always consistent or thorough because there is simply too much data in the CRAS database for a human to consume. It can be tediously boring to go through hundreds of reports. Before ACE, analysis was performed no more than once a month.

How it works In using a set of rules to go over the database that an analyzer would use, ACE finds trouble spots and determines what caused the trouble, where, and in what kind of equipment. It down-line loads data from the CRAS database every night. ACE looks to CRAS like a user generating inquiries at a dumb terminal. The next day, the user interrogates the database on ACE that consists of what the expert system has discovered. Because they can be done more often, the automated cable analyses give a more current view of the network. Hence, they are more accurate and of better quality.

The inquiries that ACE submits are based on knowledge about cable analysis. Some of this knowledge comes from existing documentation. The rest consists of heuristic strategies garnered from the experience of expert analyzers from Southwestern Bell and New England Telephone. ACE was field tested with these two operating companies and with Illinois Bell last year. The expert system is now running in at least four companies and 20 installations.

ACE runs on an AT&T 3B2/300 minicomputer under Unix. The AI portion of ACE was written in OPS4 and Lisp and is supported by code written in C. It consists of five modules that reflect the different areas of analysis with which the various operating companies were most familiar at the time of writing. The user can elect to turn on or off any of the five expert system modules.

One module, Trak, is a tracking program. After maintenance personnel (or craft) have worked on a problem, ACE can be instructed to monitor that problem to find out if the repair was effective or if the problem is getting worse. Another module, Plus3, is related to Trak. Plus3 monitors customers who have called more than three times within 30 days, generating three separate trouble tickets that the craft had thought of as already fixed.

The Found Trouble Analysis (FTA) and Trouble Report Improvement Methodology (TRIM) modules are the workhorses, the diagnostic expert systems that are used most often. Telephone companies may run one or both of these every night. TRIM looks for chronic, intermittently occurring problems that could not be found by the craft. For example, a lightning strike may have caused pinpricks in a cable, allowing rain to soak the cable with a resulting loss in sound quality. However, the problems may clear up by the time craft arrive if the cable has dried. In fact, the testing process itself may even dry the line. Sound quality improves, customers are mollified, and yet nothing has been done to fix the real problem.

The other workhorse module, FTA, looks for underlying causes of problems that have been found and repaired. However, in these cases, only the symptoms may have been fixed while the real trouble is festering elsewhere. For example, five problems in the same neighborhood were treated over a short period of time. By searching the CRAS database for patterns, FTA led to the discovery of the source of the trouble--a telephone company wiring box in chaotic disarray. Extensive jury rigging for short-term relief had turned the box into a virtual rat's nest of twisted pair. Oh, what a tangled web they wove!

As mentioned, the expertise of the various operating companies led to the creation of the different modules. For example, transient problems caused by exposure are of greater concern in the South and the West (the Sun Belt) because the networks there are still evolving and the plant is less likely to have been buried underground. Established networks such as those in the Northeast usually have more hard failures and fewer transients, although of course this is only a tendency.

Switch watch GTE's research and development arm, GTE Laboratories, has done for switches the same thing that Bell Labs has done for wiring. GTE Labs developed a diagnostic expert system, the Central Office Maintenance Printout Analysis and Suggestion System (Compass), for analyzing maintenance messages from GTE's No. 2 Electronic Automatic Exchange (EAX). The EAX is a large telephone switch that handles many thousands of lines. In such a switch, each call that is not completed generates an output message with a timestamp. These messages are collected by a Remote Monitor and Control System (RMCS). Each RMCS, which runs on a DEC PDP-11 minicomputer, can handle several switches.

Currently, as with ACE, an expert must study a long list of output messages from the RMCS to isolate faults. However, such expertise is rare. It is also lost over time, for as experts learn newer switches, they forget how to maintain the existing base of older ones.

A development group decided to incorporate the best available knowledge about the No. 2 EAX into an expert system, since that switch was to be used for several years. They worked with a switch expert from GTE Southwest and coded his knowledge into Compass.

Also like ACE, Compass runs in batch mode. The machine that contains Compass makes a telephone call to an RMCS machine, which gives Compass messages from the switches attached to it. Compass, mimicking the thought processes of a switch guru, sorts the messages into related groups. It then searches each group to find the

root problem causing the messages, tries to decide the likelihood of such a problem occurring, and makes a list of possible repairs. Finally, it reorders the problems based on how likely they are to have occurred, how easy they are to repair, and how much risk there is that a repair will interfere with service. The result is a document suggesting maintenance activity, presented in a format familiar to repair personnel. Compass also produces a "marked-up" copy of the output it received from RMCS.

Compass, implemented with KEE on a Xerox Lisp machine, is currently a field prototype. One field trial is being conducted by GTE Data Services in Tampa, Fla., and another with a site in Texas. Maintenance groups at these sites agreed to do only what Compass told them to do for the duration of the test. A number of problems were located by Compass, including one that had long baffled human experts. (In this case, the experts had been looking in an area of the switch where they had found several previous problems; Compass, however, did not share their preconceptions.) All the problems located by Compass were confirmed to be genuine problems.

GTE's expert system is being deployed in Florida, Texas, and the company's Midwestern telephone operations. As with other expert systems that work, the demand of potential users for Compass is reported to be great. Even the demonstration version has developed strong support from the GTE telephone companies. An expanded version is under development, and GTE Labs is developing additional expert systems for other applications.

Communicating consultant Perhaps the most sophisticated use of data communications in a network-diagnostic expert system is DEC's interactive program, the Network Troubleshooting Consultant (NTC). NTC can ask another program to instruct a distant node to diagnose itself.

DEC uses the expert system internally. The company has three worldwide Remote Diagnostic Centers (RDCs) for field service, in Colorado Springs, England, and France. A Telephone Support Center (TSC) group at each RDC takes calls from either internal or external customers (Fig. 5). Their Ethernet- and Decnet-related problems may include the network being down altogether or degraded, inaccurate performance. When symptoms begin to appear, the TSC tries to troubleshoot the problem.

The NTC program installed on a VAX minicomputer at an RDC acts as a back-up to second- and third-shift support personnel, who work when human experts may not be available. A TSC worker presents symptoms to NTC by entering into a dialog with the expert system. NTC eventually prescribes a remedy.

Like Bellcore's Smart, NTC asks the user questions. The knowledge base is tree-structured, so that questions narrow down the possible scope of the problem to isolate the cause. NTC's user interface has several unusual characteristics. For example, typing in a "?" lists possible ways to answer the question, while entering "why" brings up a justification of why a question is being asked or an explanation of the expert system's reasoning (how it reached its conclusion).

NTC uses networking to solve certain problems. Decnet contains a module called the network management (NM) function. While human troubleshooters can access a set of tests in NM manually to probe a remote VAX, NTC automates this test-request process and the interpretation of the results. It decides when it needs to know something that NM could find out, requests the test, and carries on the session while waiting for an answer. Meanwhile, NM checks the customer's VAX directly (if it is on the same Ethernet or Decnet network) or over a dial-up phone line.

The future In its current form, NTC is a productivity prototype. It is strictly for internal use because of the proprietary information it contains about DEC's field service operation. Not surprisingly, there have been a lot of inquiries, especially from DEC's largest customers, about whether NTC will be released as a product.

Having surveyed the field, it appears that AI techniques are a natural match for network problems. While most work has focused on expert systems, other bounty from the AI harvest, notably natural language work, is likely to spill over onto network managers' tables (see "TI AI: A different tack"). Although the field is new and most efforts are still prototypes or internal tools in large research groups, the trend is clear: AI will have a lot to offer networking.

The reverse is also true: Networking may enhance and radically change AI. Distributed artificial intelligence (DAI) will eventually have a major impact on mainstream network activities. It may even foster automation at higher levels of the organization than would otherwise be possible. A report on the development and significance of DAI will appear in an upcoming issue of DATA COMMUNICATIONS. TI AI: A different tack While expert systems research is bearing fruit for network designers and managers, another branch of AI holds great promise for users. Natural-language processing seeks to provide friendlier interfaces to computers. The extension of this idea into the area of data communications aims to simplify the interactions between the user and the network.

For example, routing in a telephone network requires complex signaling sequences, but the user of a pushbutton telephone does not have to understand any of these

codes to make a call. However, a user does have to be a bit of an expert to get around a large computer network and to use even a modest range of applications.

Texas Instruments Inc. (TI) is working with a form of natural-language processing to make its internal data communications network easier to use.

TI's private data network may be the largest in the world. Thirty-five thousand devices reside on an SNA (Systems Network Architecture) network connecting hundreds of computers in 22 major data centers. But many of the resources on this network are not being as widely used as they could be because of the syntax problem. It can be difficult to remember the rules and procedures needed to access the hundreds of services available.

To address this problem, TI has been working on a natural-language interface to network resources for about two years. The company's software, Natural Access, depends on the use of an intelligent workstation. Fortunately for the software's developers, about 17,000 such devices are attached to TI's network.

Human language as common ground Most natural-language software is written to interpret sentences typed in by users. Users, however, may not be aware of the name of the desired service or option. Therefore, TI takes a menu-based approach. Rather than forcing a user to know transaction codes or to fill in a blank template, TI's software provides menus of available options.

The four major components of the TI software are the master menu, a menu customizer, communications software, and a natural-language interface.

--The master menu is a listing of all the services that an individual user can access from a terminal.

--The menu customizer lets the user customize the master menu (adding or deleting services).

--Communications software acts as a front-end into the network. If the user wants to log on, for example, to electronic mail, the software determines which computer and which network the electronic mail software is running on and logs the user onto the desired service.

--Once connected to the service, the user may give commands in the service's native syntax. If a user does not know how to use a service, a natural-language interface provides a series of menus listing possible commands. As the user selects from the

menus, the interface builds sentences in English (or some other language) in the top window on the screen. The sentence is then translated into the command language of the application. Such sentences can be edited, stored, and regularly re-executed as macros.

Note that all of this occurs on the workstation, before an expensive log on. Thus the software accomplishes three things:

--The application interface is distributed to intelligent workstations.

--Commands are converted into the user's native language.

--A consistent interface is provided to such things as applications, operating systems, and network services.

In this way, natural-language AI techniques can assist in the use of networks for information access. A menu-based implementation such as TI's promises to change computing habits and network use. With a consistent interface, users who once dealt with dozens of services or network resources will be able to take advantage of hundreds.

So far, TI has only offered the software to a hundred or so internal users. The company believes that the software will increase the use of the network and, even more dramatically, of the data available. TI, which is providing tools for value-added resellers, expects to see a significant amount of software of this type becoming available in products over the next few years.

Photo: 1. Designet example. A sample design session is shown here using BB&N's expert system software. The monochrome (A) and color (B) displays work together. Three design stages are terminal (B) and node (C) clustering and backbone layout (D). Command box is at E. Several views can be shown at once (F).

Photo: 2. More. The network can be viewed logically (A). Automatic (B) or more manual (C) design improvement methods can be used. The latter may produce a report, as in (D). The color display is updated to reflect this (E). Based on these results, a line is selected (E) and deleted (F), which improves performance in the model.

Diagram: 3. Bilingual helper. Avant-Garde's expert system, Net/Adviser, keeps its model of the user's network up to date with alerts from several control tools. With each change, the Prolog engine checks its rules for advice, which is held until the user requests it. Special code links the rule base to the network model. Diagram: 4. Relief. SCC operators with tough switch problems may turn first to the Smart expert system. Controllers at the SCC may not even see

alarms that React handles. Diagram: 5. Go out and check. If, in talking to the support person, the NTC expert system needs a test run on a remote VAX, it asks the NM function to request it.

Document dcom000020011119di7100061

Search Summary	
All of these words	
At least one of these words	
None of these words	
This exact phrase	AI carves inroads: network design, testing, and management
Date	All Dates
Source	All Sources
Author	All Authors
Company	All Companies
Subject	All Subjects
Industry	All Industries
Region	All Regions
Language	English
Results Found	3
Timestamp	22 October 2018 10:41 PM