NASA CONTRACTOR REPORT 177346

The Application of Artificial Intelligence Techniques to Large Distributed Networks

Ralph O. Dubayah
Professor Terrence R. Smith
Jeffrey L. Star

CONTRACT NCA2-OR680-401
April 1985

NASA

NASA CONTRACTOR REPORT 177340

The Application of Artificial Intelligence Techniques to Large Distributed Networks

Ralph O. Dubayah
Professor Terence R. Smith
Jeffrey L. Star

Prepared for
Ames Research Center
under Contract NCA2-OR680-401

April 1985

National Aeronautics and Space Administration

Ames Research Center
Moffett Field, California 94035

I. Introduction

This report represents part of the tangible outcome of a small pilot investigation into the applicability of artificial intelligence techniques to large distributed data processing networks. A first goal of this investigation included a review of both the relevant AI literature and of NASA requirements in order to educate the investigators. A second goal involved the transmission of the main findings of the investigators to NASA personnel in terms of a set of three lectures and this final report.

NASA is currently considering the construction of large distributed data processing networks. Such systems involve distributed data, distributed software and distributed hardware. The coordination of such systems is a difficult problem of vital interest. It is apparent that the techniques of AI are likely to prove of great value in such a task of coordination. In this report we examine some aspects of the applicability of AI particularly to the Pilot Land Data System (PLDS) program. In particular we examine the concept of distributed problem solving systems (DPSS) to such tasks of co-ordination.

The topics discussed in this report include:

1. PLDS and its data processing requirements
2. the applicability of expert systems to co-ordinating PLDS
3. the concept of DPSS as a unifying structure for coordinating distributed systems of interest to NASA
4. AI problem solving paradigms
5. distributed problem-solving systems
6. DPSS control strategies
7. query processing
8. distributed databases
9. applicability of DPSS to NASA systems.

We also provide an appendix on AI techniques.

II. Pilot Land Data Systems and Their Requirements

1. Requirements of PLDS

The NASA Pilot Land Data System Programs (PLDS) represent an attempt to deal with the problems of distributed data and distributed processing. The purpose of the PLDS is to "establish a limited scale distributed information system to improve communication, data access, and data sharing."

The easiest way to understand the PLDS concept is by examining its underlying data analysis requirements. Such a system, would require the following characteristics:

a. the ability to move data sets rapidly from data management systems to locations where research is being conducted.

b. the ability to register, calibrate and modify data sets to standards rapidly with minimal intervention by the scientists.

c. the ability to perform complex processes on data sets in near real time, both locally and remotely.

d. the ability to communicate research data and technical information between scientists locally and remotely in near real time.

e. the ability to have access to all software and hardware tools necessary to support a complete research project.

2. Configurations of a PLDS

A Pilot Land Data System (PLDS) is now in the planning stages, with implementation beginning in FY 85. PLDS is a complicated data system, involving many more networked nodes, distributed around the country, as well as a heterogeneous distributed database.

Based on the early discussions at the PLDS planning workshops (The Pilot Land Data System: Report of the Program Planning Workshops, NASA Technical Memorandum 86250, July 1984), a preliminary topology for the network is:

```
          ::::::::::: NASA satellite network :::::::::>>
             //
          ==//====== long-haul packet network ==========>>
           // //
          // // .::: telephone // dial-up network :::::::::>>
         // // //            //  //                //
      ..........          ..........          ..........
      :gateway .          :gateway .          :gateway  .
      : /      .          : /      .          : /       .
      :local net .        :local net .        : local net  .
      : /    /   .......  : /    /   ........ : /     /    ....
      :vax   workstation: :IBM   workstation. :work    work    :
      : /               : : /               . :station station :
      :workstation      : :workstation      . ..................
      :.................: .....................  Small University
         NASA Center        Other Major Node
```

Redundant communications channels of different performance and cost interconnect nodes of different characteristics. For planning purposes, nodes of at least three different kinds have been distinguished. Level 1 users have large computer resources and a high degree of sophistication. These might include the major NASA centers, with sophisticated networking capabilities, mainframe CPU power, and large staffs. Level 2 users could include smaller laboratory groups and other state and federal agencies, with minicomputers or simple video displays, without access to high speed or dedicated networks.

Communications for the network will probably include a number of methods:

For conventional dial-up services, several routes are available:
via Bell, MCI, SPRINT, etc.: These are low investment circuits, with a high cost per bit, 300/1200 BPS Bell 103/212a compatible asynchronous communications. Particularly at higher speeds, these are not always reliable over the long distances of the PLDS network.

Value-added common carrier network: packet switched with 300/1200 BPS interface. Tymnet, Telenet, etc. Lower cost per bit, computer-oriented service.

Leased line:
1200 BPS and higher. Leased and conditioned from common carriers. High fixed cost of ownership, lower potential cost per bit if the service is used at a high enough level.

Satellite:
Expensive but cost independent of distance. 56 KBPS to 50 MBPS. Long lead times required for installation.
Necessary service on the network include:

Catalog of available data

Inventory of on-line data

sophisticated retrieval/archival support

uniform user interface

facilities for data manipulation

consistent storage

AI Problem Solving Paradigms

There are currently several problem solving paradigms in common use in AI research. These are: describe and match, goal reduction, constraint propagation, search, means-ends analysis, generate and test, and rule-based or production paradigms.

The describe and match method of problem solving is useful for solving geometric problems and the like. We first describe rules that apply to a given situation. We then try to match the rules description. For example, we may describe a rule as "place the small circle inside the larger circle." We then try to match this rule.

The process of achieving a goal by first achieving subgoals is known as goal reduction. Goal reduction is an important paradigm for DPSS. A goal tree is created with subgoals as nodes on this tree. Picking the path through this goal tree is a search problem.

The constraint paradigm uses knowledge of the application domain to constrain the possible solution sets. For example, if trying to find the vertice of a cube on an image, the solution set can be constrained to intersections of three lines.

Search is an ubiquitous problem in AI. It is used for route finding, problem reduction, rule based problem solving, theorem proving, learning, and wherever one choice leads to another. There are three major groupings of search algorithms: a) those that find some path to the solution b) those that find the optimal path c) and those that deal with adversaries (game playing). Some specific algorithms that find some path through the goal tree are depth first, breadth first, best first, beam search, and hill climbing techniques. Some optimal path algorithms are branch and bound, A*, and dynamic programming. Minimax, alpha-beta pruning, heuristic pruning, progressive deepening are game playing search procedures.

Another AI problem solving paradigm is means-ends analysis. Here, procedures are selected by their ability to reduce the observed difference between the current state of the problem and the solution state. For example, if procedure 1 gets us closer to the goal than procedure 2, we pick procedure 1.

The generate and test paradigm is a two step process. The first step is the enumeration of possible solutions (the generation phase). The second is the evaluation of each proposed solution (the testing state). The generation step may produce a tremendous number of possible solutions, all of which could never be tested. The key, then, is to use informed generation.

Rule based paradigms are the basis for production systems. These use collections of if/then rules to solve problems. There are forward and backward chaining systems.

III. Distributed Problem Solving Systems

1. The Concept of a DPSS

While the application of AI in terms of ES at various nodes in the system is of advantage, we believe that a more unified approach to the design of such a system is justified. The approach that we propose involves the concept of a distributed problem solving system (DPSS). In particular the ES's discussed above should really be considered as components of such a DPSS.

A DPSS network is a distributed network of distinct, semi-autonomous problem solving nodes which can cooperatively interact among themselves to solve a single problem. AI programming is an integral part of the system because node activation and interaction is controlled by modules having knowledge of the problem solving domain, and the problem solving network hardware and software. A major strength of DPSS techniques is the integration of distributed hardware and software. Spatially distributed hardware can be integrated through intelligent control software. Hardware redundancy among sites is avoided with such control software while parallel and concurrent processing is promoted.

Intelligent control software can also integrate distributed applications software, such as data base management systems, again avoiding redundancy and promoting parallel processing. The overall effect of this integration through DPS is to create a more powerful total system in which the complexities of the integration are transparent to the user.

2. DPSS and Their Characteristics

As stated earlier, a DPSS network is a distributed network of distinct semi-autonomous nodes which can cooperatively interact among themselves to solve a single problem. DPSS can be viewed as a four-phase process: problem decomposition, sub-problem distribution, sub-problem solution, answer synthesis. Why use DPSS techniques? The advantages include:

a) Reductions in the complexity of computation through the decomposition of processing

b) Modularity

c) Increased efficiency through parallelism and concurrency

d) Graceful degradation of response

a) Reduced Complexity

DPSS techniques can reduce the complexity of computation through the decomposition of processing (task sharing). Central computation for complex problems is costly in memory and time and prohibits certain types of computational activities. Problem distribution allows work by different processors on non-overlapping segments of the problem to proceed at the same time. Further, each processor can then be limited in scope, reducing the complexity of each node. This implies specialized expert systems of limited input domain. This is an important characteristic since complexity of computation can be an exponential function of input space size.

b) Modularity

In a DPSS, modularity is inherent. Modular systems can evolve and adapt easily. Local changes in the system do not affect the overall structure of the system. If the structure of the network reflects the nature of the problem, the rate of change of the external framework should be low, even if subjected to high LOCAL rates of change.

c) Increased Efficiency Due to Parallelism

Task decomposition allows for parallel processing. Efficiency is also increased through the twofold process of optimization and replication.

Once a processor is optimized for a particular subtask, it can then be replicated as often as necessary to avoid processing bottlenecks. If it is not needed, the processor is simply not replicated, thus avoiding redundancy. Replication, therefore, does not imply redundancy.

d) Graceful Degradation of Response

DPSS allows for the graceful degradation of response in the event of input data noise or failure, or the failure of nodes in the network. This greatly enhances the utility of the entire system and drastically reduces "down time" for practical purposes.

3. DPSS differ from distributed processing

One common disconception is that distributed processing and distributed problem solving are essentially the same thing. This is incorrect. Distributed processing systems typically have a network of machines executing multiple, widely disparate tasks concurrently. A good example of this are the remote tellers (Versateller, ReadyTeller) that many banks now have in use. Interactions among tasks is done mainly because of shared access to physical/informational sources.

In DPSS, there is generally only a single envisioned for the system. Nodes have information about the distribution of network components, subtasks, and data, and act accordingly. Task interaction is, in general, a necessary part of the problem solving system because nodes must share intermediate results. This communication among nodes, however, is one of the large problems of DPSS. It is far cheaper to compute than to communicate. Thus, internode communication must be limited to avoid large bandidths. This forces the nodes of the network to be loosely coupled and to use knowledge of the system to carefully plan their communication acts taking into account the planning and inference abilities of the receiving agents. Exactly how the nodes are going to communicate and interact is determined by the control strategies used by the DPS system.

4. DPSS Control Strategies

Give the above discussion, how can different problem solving procedures interact and how does a DPS system know which of several procedures to choose? In other words, how are control decisions made? Winston (1981) asks the following questions about control choices:

a) Where is knowledge about procedures stored?

b) What process decides which procedures act?

c) How are computational resources allocated?

d) What kinds of procedures exist?

e) How and to what degree should procedures communicate?

It is the answers to these questions that determines how the DPS will effectively run.

5. Query Processing and DPSS

Query processing is an important component of a DPSS. Issues of concern here include parsing, representational languages, and natural language interfaces. The decomposition of a query into a set of base components is known as parsing. The routing and scheduling of these components to various problem solving nodes promotes parallel processing. Natural language and graphics front ends enable user friendly interfaces.

Distributed query processing is different from nondistributed query processing. First, there is a processing delay and a communication cost among the sites involved in the query. Secondly, there is the opportunity for parallel processing as discussed above. Algorithms for handling distributed queries usually involve a) the definition of a family of strategies which can be applied to compute the result of a query. b) an optimization to choose the least costly member of this family--query optimization.

6. Current Distributed Query Processing Systems

One such system is IBM's System R*. R* features a SQL compiler which chooses an access plan for a given query by generating a set of possible strategies for that query and assigning a cost to each plan. It then picks the cheapest plan. The cost is figured as:

COST = I/O cost + CPU cost + communication cost

Because each plan must be determined in advance, the improvement in query strategies may not be worth the additional planning cost.

Another query processor is System Development Corporation's MERMAID system. Query optimization is achieved through a five phase process of a) closure b) site selection c) local reduction d) global reduction and e) assembly.

In time, however, every DPSS will require a Knowledge Based Query Processor (KBQP). KBQP uses knowledge of the DPSS application and databases to transform the query into equivalent statements which are more efficient to process. When designing KBQP the following are important: a) the kinds of knowledge that should be included in the knowledge base, b) how this knowledge should be expressed, c) the kinds of transformations that can exploit this knowledge to improve query processing, d) the way in which the system as a whole can be organized in the presence of large, intricate knowledge bases.

7. Distributed Data Bases

The accessing of heterogenous, distributed data bases is critical for satisfying NASA's future needs. DPSS must deal with this problem efficiently to be effective. Fortunately, all of the issues related to heterogeneous DBMS are the same as those for homogenous DBMS except for a substantial translation problem. The goal of a heterogenous, distributed data base management system (HDDBMS) is to give the user an integrated, yet transparent, view of the various data objects and enable the integration and sharing of these data among loosely coupled nodes on a local area network (LAN), between LANs or on a long haul network.

A HDDBMS is a system in which the nodes of the DDBMS differ in data model and data access. Intelligent front ends have been created which integrate existing DBMS's to provide the user with a single language to manipulate data in several data bases that are stored on different computers under different operating systems. Some current front ends are the Mermaid system by SDC, MULTIBASE by CCA, and SIRIUS-DELTA by INRIA in France. Some current DDBMS are SDD-1 by CCA, distributed INGRES of CAL, Encompass by Tandem, and R* by IBM.

IV. Expert Systems and Coordination of PLDS

1. The Little Washita River Basin Real Time Hydrologic Model

In order to provide a background for discussing the problems that would arise for practical users of such a PLDS, we provide a concrete example of a distributed real-world system that involves data from a variety of sources.

The Little Washita River Basin has been proposed as an area for evaluating various hydrologic models as well as the PLDS that would be developed to link the various participating organizations in the project. The model, as developed, uses daily geostationary satellite data, weekly and monthly data from other sensor systems, geophysical data, etc. to estimate streamflows, distribution of storage throughout the watershed and REAL TIME fluxes of various hydrologic elements.

The model was designed to run in real time, yet it cannot be tested due to its inability to deal with distributed data sources (etc). It is currently impossible to obtain all the data from the various sites appropriately formatted and processed to perform real time evaluation. The researchers involved have expended large amounts of creative energy designing a model that no computer can currently run. This is not because the model is too large or the data sets are too massive, but because the data sets cannot be accessed and processed in a timely manner. This situation prompted the scientists to lament "The utility of the Model and the validity of some of its formulations have not been examined because it has been impossible to obtain and interface many of the critical data elements."

What are some of these required model data elements?

streamflow -	Maryland, USDA maintained gaging stations
water quality -	Southern Great Plains (SGP), Beltsville data files
rainfall -	point and spatially integrated isoheytals from NOAA Severe Storm Center in Oklahoma, USDA at SGP, Beltsville
climatological -	NOAA: temperature, humidity and winds

terrain data -	digital elevation files from USGS, USDA
soils -	USDA at SGP, Beltsville
satellite data -	historical MSS, current MS, TM, AVHRR, GOES, etc.

Looking at the diversity of these key elements, it is easy to see why this model cannot be run in real time: The data sets are stored in various data base management systems running under different operating systems on different CPU's. The data types, formats, scales and data storage models may all be incompatible. There may be required distributed applications programs. Accessing this distributed data, storing it, processing it for scale, format, rectification, etc., would require an extremely complex program, one that would then be only site specific for the particular CPU's, DBMS's, data formats, etc. and would still not run in real time.

2. The need for System coordination

The example of the Little Washita Basin project indicates well how the PLDS could require a controlling system of some power in order to allow the users of such a system to overcome the problems involved in working with distributed data, distributed software and distributed hardware. In particular such a controlling system would need a good deal of knowledge about the different subsystems that it was required to interface and a significant degree of problem-solving ability.

3. Expert systems and coordination

Such a communication, however, is by itself insufficient to achieve the five data analysis requirements listed above. The creators of the PLDS have realized this, and have proposed three artificial intelligence (AI) systems to aid in the achievement of various requirements. The first of these is a natural language query processor. The second is an expert system for data base management, system controls, and operations and third is an expert system for performing complex data search and information detection, identification and cataloging.

There is little doubt that such expert systems (ES) should be incorporated into such a system, and we can immediately expand these three ES in terms of both numbers of functions and numbers of ES. Let us consider a few examples.

A network interface expert knows how to send error-free messages to another node. This includes a number of a relatively straight-forward functions: least-cost route selection based on time of day and service requirements, packet construction where required, error detection and retransmission, baud/word format details, store and forward via intermediate nodes if portions of the network are down or busy. This expert should be very portable--anyone on the system with any programmability could use this expert.

A Data Catalog expert knows how to access data catalogs in various locations around the network, and knows where data catalogs are located. This expert can act as a front end to the different DBMS's, and translate between different DBMS query languages. This last part--translation between different query languages--is a relatively simple problem, since the contents of a query are relatively standardized. In other words, we might expect that while the syntax of a database query is different between systems, the semantics are quite regular.

A Data Format expert knows, based on the source of some data, what the format of the data is (for example with satellite imagery, whether the data is band interleaved by line), and understands what to do to reformat the information so that it arrives at a given node in a locally useful form (i.e., DEC vs. IBM byte format within words).

A Geography expert understands three different areas: map projections, place names and state/county/census boundaries, and latitude/longitude vs. satellite scene identification. The knowledge base for this expert is available to some extent from the computer-compatible holdings of U.S. Geologic Survey.

A Rectification expert knows about rectification and registration procedures. This expert knows the characteristics of particular algorithms and where the details of these algorithms are located. This expert knows how to ask "how accurate do you need it?" and how to assist the specification of control points (in part, by using the geography expert). This expert would be a good pilot for implementing distributed problem solving.

On another axis, consider that any of these experts may be centralized, duplicated at many nodes, or distributed. For example, the data catalog expert may be important to the efficient operation of any node with a computer which is using the network often. If this is the case, it may be efficient to duplicate the expert system at each node, saving communications costs but causing currency problems as well as the costs of duplicate information storage. At the other extreme, there may be a single data catalog expert at one node on the network, which is accessible by anyone on the network. This would cause more traffic on the network but a relatively simple task to implement. A third alternative would be to decentralize the expert: at any node, the knowledge base and rules of inference would be tailored to the kinds of queries made locally. To take one concrete example, the data catalog expert at a particular node may be able to translate queries in a number of foreign database query languages into the local dialect, as well as translate local query language into one or two of the most frequently needed foreign dialects.

4. DPSS and the Washita River Project Example

Concerning the case of the Little Washita River Project, for example, a DPSS would first decompose the task "run the hydrologic model" into subtasks and queries, and then route them to appropriate sites at appropriate times determined by a knowledge-based system. The data would be accessed and preprocessed at each site and the answers synthesized into a final answer. The great advantage of DPSS is that the model software can be distributed among several sites, thus freeing any one CPU from being bogged down by running the entire model. A DPSS that can handle problems such as those of the PLDS requires expertise in the following areas:

- AI, expert systems, machine learning
- query processing
- distributed data bases and DBMS
- distributed control procedures, concurrency control
- parallel processing
- distributed software applications
- networking and distributed hardware architecture
- distributed processing

Before we look at some of these areas as applied to distributed problem solving, problem solving in general should first be discussed.

V. The Applicability of DPSS to NASA Systems

There are tremendous potential applications for DPSS within NASA in both the short and long term. The utility of DPSS as applied to one system--the PLDS--has already been shown. The underlying data analysis requirements of the PLDS are such that the hardware and software components of the PLDS could be fit into a DPSS. It would therefore be short-sighted of the PLDS planners to construct a system which has no upward compatibility with DPSS. The individual expert systems of the PLDS should be viewed as building blocks which alone are of limited value, yet linked as through a DPSS, are extremely valuable.

Expert systems and the technology behind them, are proliferating a a rapid pace. However, no institution has yet implemented a plan which utilizes this technology in its most powerful form: as DPSS. NASA can become a leader, <u>not</u> in data processing, <u>not</u> in expert systems, <u>not</u> in DBMS, but in knowledge acquisition and dissemination by creating a DPSS. Its scientists, researchers, and administrators will operate more efficiently and effectively, attacking problems once thought unsolvable.

Eventually, the cost of research will go down due to the lack of redundancy in hardware and software among NASA research centers and their institutional affiliates. These costs will be distributed among <u>all</u> the locations rather uniquely for it will no longer be necessary for individual sites to have massive computing power, nor extensive data bases and DBMS. NASA scientists will no longer have to spend weeks waiting for data only to find it will require more weeks of work to construct programs which can massage the data into a usable form. Freed from these computing tasks, they will be able to spend more time doing what they were trained to do--science.

BIBLIOGRAPHY

Carlesi, C., Ricciardi, Thomas, "Data Distribution Criteria and Query Processing Strategy in a Small Distributed Environment," in Distributed Data Bases, edited by Delotel and Litwin, North Holland Publishing Company, 1980.

Corkill, Daniel, Victor Lesser, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," The AI Magazine, Fall, 1983.

Date, C.J., "An Introduction to Database Systems," Addison-Wesley, Reading, MA, 1983.

Epstein, Robert, Sonebraker, Wong, "Distributed Query Processing in Relational Data Base Systems." Proceedings of ACM SIGMOD International Conference on Management of Data, June, 1978.

Franklin, Janet, "Draft Final Report of the Pilot Land Data System Program Planning Workshops," NASA, 1983.

Freeman, M. et al., "Logic Programming Applied to Knowledge Based Systems, Modeling and Simulation," Conference on Artificial Intelligence, Carnegie-Mellon University, August, 1982.

Gevarter, W.B., "An Overview of Artificial Intelligence and Robotics" NASA Technical Memorandum 85836, 1983.

Ragan, Robert M., "A Test of Real Time Remote Sensing Hydrologic Modeling," NASA, 1983.

Rothnie, J.B., et al., "Introduction to a System for a Distributed Databases (SDD-1)." ACMTODS, vol. 5, no. 1. March 1980.

Templeton, M., D. Brill, A Hwang, I. Kamney, E. Lind, "An Overview of the MERMAID system - A Frontend to Heterogenous Databases," in Proceedings of ACM SIGMOD, May, 1983.

Winston, Patrick Henry, Artificial Intelligence, Addison-Wesley, 1984.

Yu, C., C. Chang, M. Templeton, D. Brill, E. Lind, "On the Design of a Query Processing Strategy in a Distributed Database Environment," in Proceedings of ACM SIGMOD, May, 1983.

1. Report No. CR - 177346	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle The Application of Artificial Intelligence Techniques to Large Distributed Networks		5. Report Date April 1985
		6. Performing Organization Code
7. Author(s) Ralph Dubayah, Terence R. Smith, Jeffrey L. Star		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address Geography Department / Community and Organization Research Institute, University of California, Santa Barbara, CA 93106		11. Contract or Grant No. NCA 2-OR680-401
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address Point of Contact: William Likens, Ames Research Center, Moffett Field, CA 94035 (415) 694-5595 or FTS 448-5595		14. Sponsoring Agency Code 656-13-50

15. Supplementary Notes

Point of Contact: William Likens, MS 242-4, Ames Research Center, Moffett Field CA 94035 (415)694-5596 or FTS 448-5596

16. Abstract

Major NASA initiatives revolve around the problems of data accessibilty and transfer of information, including the Land Resources Information System Pilot, are structured as large computer information networks. The goals of these pilot efforts include reducing the difficulty of finding and using data, reducing processing costs, and minimizing incompatibility between different data sources for researchers both within and outside NASA.

Artificial Intelligence techniques have been suggested as a means of achieving these goals. In this report, the applicability of certain AI techniques are explored in the context of distributed problem solving systems and the Pilot Land Data System (PLDS). The topics discussed include PLDS and its data processing requirements, expert systems and PLDS, distributed problem solving systems, AI problem solving paradigms, query processing, and distributed data bases.

17. Key Words (Suggested by Author(s)) Artificial Intelligence, distributed problem solving, Pilot Land Data Systems, expert systems, distributed data bases		18. Distribution Statement Unclassified - Unlimited Subject category 63	
19. Security Classif. (of this report) none	20. Security Classif. (of this page) none	21. No. of Pages 12	22. Price*