DYNAMIC PLANNING FOR THE CONTROL OF DISTRIBUTED PROBLEM SOLVING

J. D. Yang, M. N. Huhns & L. M. Stephens
Center for Machine Intelligence

USCMI Technical Report 83-10
June 1983

Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208

DYNAMIC PLANNING FOR THE CONTROL OF DISTRIBUTED PROBLEM SOLVING

Ju-Yuan D. Yang, Michael N. Huhns, and Larry M. Stephens
Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208
(803) 777-4195 CSNET: huhns @ scarolina

AI Topic: Problem Solving and Inference

ABSTRACT

A distributed problem solving approach offers the advantages of increased real-time response, reliability, and flexibility, and lower processing, hardware, and software costs. In general, a distributed problem solving system consists of multiple processing nodes with one or more expert systems at each node. A node organization is designed to support the aforementioned approach. Some of the major components in the node organization are a database of meta-knowledge about the expertise of its own expert systems and those of the other processing nodes. These information are gradually accumulated during the execution of problem solving processes, a dynamic planning ability which guides the problem solving process in the most promising direction using the actual (if available) or estimated information, and a question-and-answer mechanism for handling the internode communication. This paper presents a description of the node organization. A logic design example is presented to demonstrate the functions of this system.

INTRODUCTION

Recent developments in processor fabrication technology and computer communication technology have reduced the cost of computing elements and communication among processors to a level where distributed processing systems are practical. In recent years, there has been growing interest in distributed problem solving systems. Generally speaking, such a system consists of multiple processing nodes with one or more expert systems at each node. The advantages of a distributed approach to problem solving are increased reliability and flexibility, enhanced real-time response, lower communication costs, lower processing costs and reduced software complexity. The processing nodes cooperate in the sense that no one of them has either sufficient expertise or complete information to solve a complete problem; mutual sharing of the task load and interchanging of information is therefore necessary to produce a solution.

Several distributed problem solving systems [1,2,3] and one testbed [4] have been developed. The approaches taken include a negotiation mechanism in the contract net [1], and a functionally

accurate, cooperative approach [2]. Some of the important issues in those systems have been identified as system control, internode communication, and problem partitioning.

A group of human experts working together to solve a large task is a familiar metaphor for a distributed problem solving system. Of interest to our problem in examining the operation of human experts are: (a) The meta-knowledge they use to make judgments about acceptance and estimation of an incoming task and who or where to submit unsolved subtasks. (b) The way in which they make necessary plan adjustments, or even change to a new plan, as more accurate information is gathered to lead the problem solving process in the most promising direction. (c) The way in which they communicate with each other to solve the entire problem.

Based on these observations, We developed a framework [8] in which each node is equipped with a dynamic planning ability, an internode question-and-answer mechanism [9], and meta-knowledge to attack those issues mentioned above, maintain global coherence, and efficiently use the available knowledge sources.

NODE ORGANIZATION

Figure 1 represents the organization of a processing node. Each node consists of a front-end processor (including a receiver and a transmitter), a planner, a scheduler, a solver, a blackboard, meta-knowledge, and knowledge sources.

Two types of meta-knowledge are available to the processing node; each type is organized according to a frame-like structure. One type stores the names of problem areas of interest to this node as well as corresponding problem decompositions and result synthesis mechanisms which its own expert systems can provide. The other type represents its estimates of the beliefs and capabilities of other processing nodes. The motivation for using meta-knowledge is to allow a high-level reasoning process to be implemented in the internode communication and planning processes.

Problems within the areas of expertise of each processing node are accepted by the receiver from a message channel, depending on an attached address and the meta-knowledge which is available. After a problem is accepted, a frame is written on the blackboard to keep the information associated with the problem. The frame has several slots to store such information as an abstraction of the problem, an associated data and specifications, a solution plan, a result synthesis mechanism, and a distribution list for results.

A planner generates a problem solving plan for each newly received problem. For an existing problem, it dynamically makes plan adjustments to lead the problem solving process in the most promising direction as more precise information is gathered. A detailed description of this mechanism is given below.

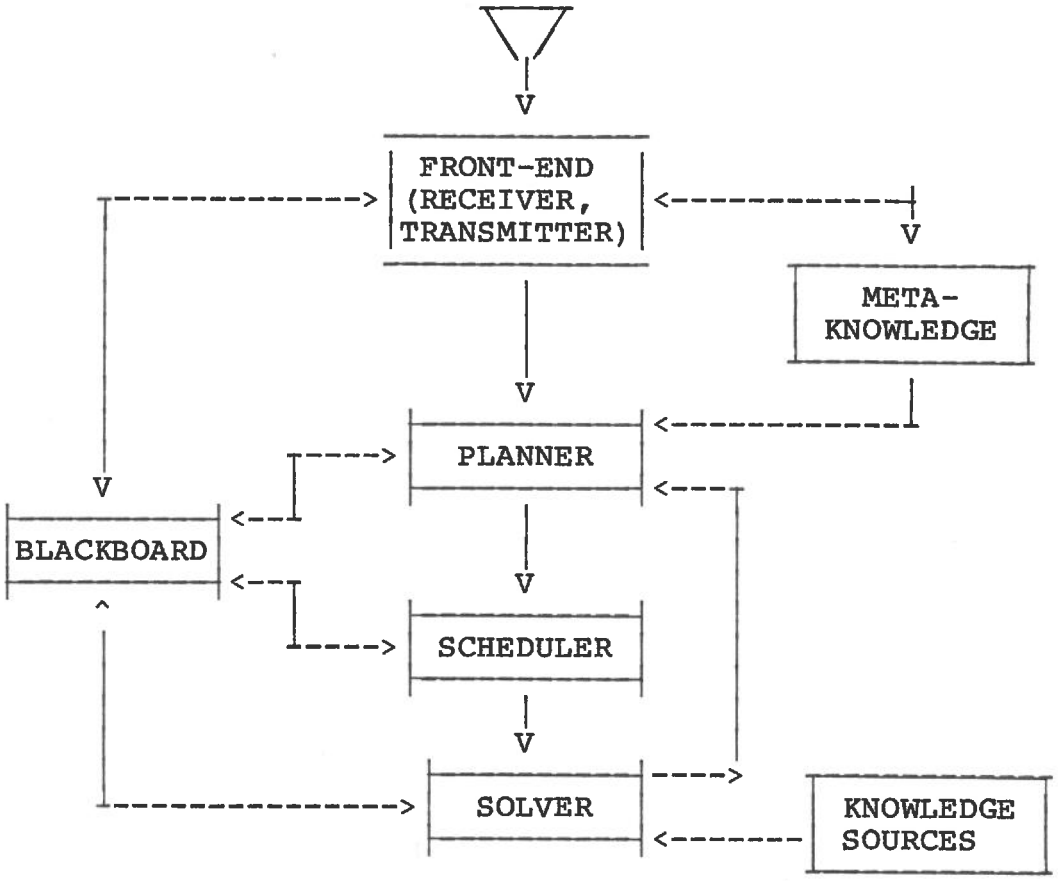


Figure 1. Node Organization.

All planned problems are ordered by the scheduler according to their importance in the problem solving process, the availability of results to their subproblems, and knowledge source requirements. The ordered problems are executed according to their priorities as assigned by the scheduler. The necessary knowledge sources are initiated, and the problem solving plan is deducted. This process is repeated until no further deductions can be made. A final result is stored into the corresponding frame on the blackboard, if the problem is completely solved; otherwise, the problem solving plan is updated and intermediate results are stored. The unsolved problem is then sent back to the planner to make further plan adjustments.

The transmitter is responsible for directly transmitting or broadcasting available results, as well as any unsolved subproblems which are outside the capability of this processing node.

PLANNING MECHANISM

The goal-directed planning used in STRIPS and ABSTRIPS [5] has been proven to be an efficient mechanism. On the other hand, the data-driven control mechanism developed for the Hearsay-II

speech understanding system [6] is regarded as an effective control mechanism for problem domains in which each processing node has only a partial view of the problem and erroneous data. A combination of goal-directed and data-driven mechanisms is used by the planner presented in this paper to dynamically guide the problem solving process in the most promising direction and maintain global coherence. The problem decomposition and result synthesis meta-knowledge about problems of interest and beliefs about other nodes are needed in this process.

The planning process begins with a goal-directed mechanism. It decomposes an incoming problem by using the appropriate problem decomposition expertise (this is domain dependent) which is available in the meta-knowledge data base. An AND/OR graph-like problem solving plan is generated after this process. In oder to have efficient use of knowledge resources (KS's) and communication channel capacity, a focusing control mechanism similar to GODDESS [7] is used to select the most promising problem solution plan from a number of possible choices in terms of several parameters. The parameters are domain dependent, and may include values for result confidence, data reliability, and solution cost.

The focusing control mechanism constructs the most promising problem solution plan by applying a problem dependent evaluation mechanism to every node in the corresponding AND/OR graph-like problem solving plan. For a leaf-node subproblem, its criterion is generated by the evaluation mechanism in terms of either actual parameter values (if available), or estimated values stored in the meta-knowledge database, and returned to its parent node for another level of solution plan selection. For a non-leaf subproblem (either an AND or OR node), its criterion is calculated based on the criteria of its son-node subproblems according to the provided evaluation mechanism. It may be the biggest, or the smallest for an OR-node subproblem, or summation, weighted summation, or the biggest for an AND-node subproblem. Hence, while the focusing mechanism provides a framework for the dynamic planning, it is the responsibility of the user to supply the domain dependent evaluation mechanism and choose the appropriate parameters.

Initially, the most promising problem solution plan is chosen based on the information stored in the meta-knowledge database. As the problem solving process proceeds, more precise information is gathered and used to select the next most promising problem solution plan, as well as to update the meta-knowledge database. This process is repeated until the problem is solved. Since the content of the meta-knowledge data base becomes more accurate as the system is used, the system should be able to generate a better problem solution plan within a few planning cycles.

INTERNODE COMMUNICATION

The internode communication is structured into a question-and-answer process incorporating meta-knowledge. From the point of view of a transmitter, a problem is transmitted as a question. When a question is received by a receiver, a task is generated whose goal is to produce an answer for that question. An answer could be either a result or a hypothesis, depending on the question.

A transmitted problem or question may be broadcast or may be addressed to a particular node. A node's acceptance or rejection of a question depends on its attached address and the scope of expertise available at this processing node. Questions and answers are directly transmitted or broadcast to appropriate processing nodes based on both the related information in problem frames and beliefs about other nodes in the meta-knowledge data base.

A common language is needed to specify the contents of the communication messages. The system uses only two kinds of messages: one for questions (task announcements) and the other for answers (result reports). The message format is like the one used in the contract net [1] in that it is composed of a number of slots that specify the kind of information needed in that type of message. A question message has four main slots: 1) message type, with question as its value; 2) message name, with an identification attached by the sender; 3) message abstraction, a brief description of the question written in common language; and 4) input, a list of associated data. An answer message also has four main slots: 1) message type with value answer; 2) message name and 3) abstraction, the same as in the question message; and 4) output, a list of results or hypotheses replying to the question.

Although this mechanism is much simpler than the negotiation process used in the contract net [1], we believe it still has the ability to control the quantity and direction of messages flowing through the message channel.

AN EXAMPLE

A logic design problem of implementing the sum of the least significant bit of a carrier look-ahead adder is designed to demonstrate the operation of the proposed system. The system in this example consists of three processing nodes. Their meta-knowledge database and available devices are shown in figure 2. A task is given to the system as a question and for this example the associated input data is a set of simple logic equations. Propagation delay is chosen as the criterion to compare the different kinds of implementations. A task can be either directly given to a particular processing node or broadcast to the whole system.

NODE NAME	INTEREST PROBLEM ABSTRACTION	RESULT SYNTHESIS PROCESS	PROBLEM DECOMPOSITION PROCESS	CRITERIA (DELAY ns)
1	logic-design	rs-mux	dec-mux	nil
	1-d-mux	rs-mux	dec-mux	14
2	logic-design	rs-nad	dec-nad	nil
	1-d-nand	rs-nad	dec-nad	20
3	logic-design	rs-aot	dec-aot	nil
	1-d-aot	rs-aot	dec-aot	25

(a) Meta-knowledge databases describing each node's expertise

NODE NAME	PROBLEM ABSTRACTION	CANDIDATES	CRITERIA
1	1-d-nand	node2	10
	1-d-aot	node3	15
2	1-d-mux	node1	10
	1-d-aot	node3	18
3	1-d-mux	node1	10
	1-d-nand	node2	12

Note:
1-d-mux stands for logic design with multiplexers.
1-d-nand stands for logic design with NAND gates.
1-d-aot stands for logic design with AND, OR, and OR gates.

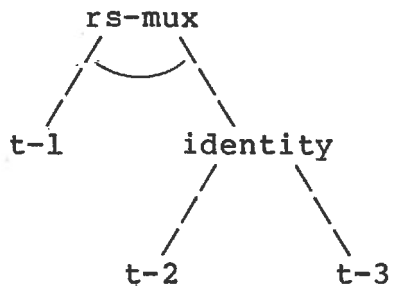
(b) Estimates of the abilities of other processing nodes.

NODE NAME	1	2	3
AVAILABLE	74153	7400(10ns)	7404(10ns),7432(4ns)
DEVICE	(14ns)	7410(10ns)	7408(10ns),7411(8.2ns)
		7420(10ns)	

(c) Available logic devices.

Figure 2. Domain dependent knowledge needed in logic design example.

In this particular example, the task shown in figure 3 is given to node one. Since the task is within the area of this node's expertise, the corresponding problem decomposition process takes place, and a problem solving plan, (t-1 (t-2 t-3 v identity) ^ rs-mux), is generated. The corresponding AND/OR graph-like problem solving plan is shown in figure 3. The task frames of the subtasks t-1, t-2, and t-3 are also shown in figure 4. Based on the information stored in node one's meta-knowledge database (15ns for t-2 and 10ns for t-3), the most promising problem solution plan is chosen as (t-1 t-3 ^ rs-mux). Among the kernel subtasks of the solution plan, t-1 is solvable, and t-3 is unsolved, so subtask t-3 is broadcast through a message channel to other processing nodes in the system.



(t-1 (t-2 t-3 v identity) ^ rs-mux)

Note:
t-1, t-2 and t-3 are leaf subtasks.
identity and rs-mux are result synthesis processes.

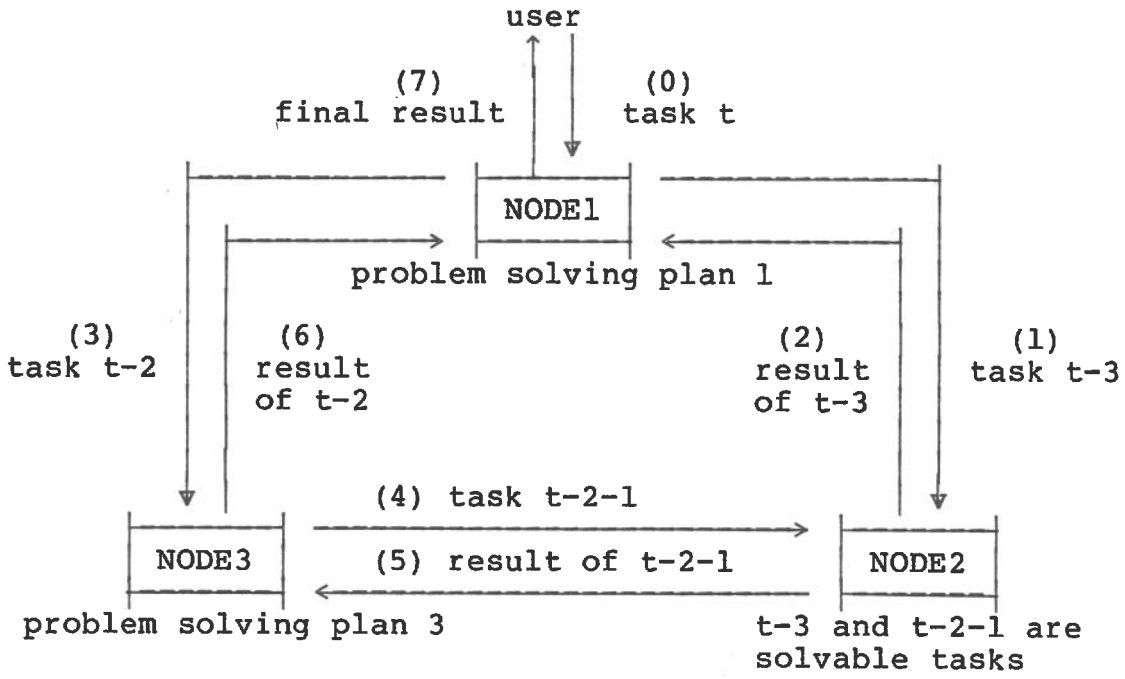
Figure 3. A problem solving plan.

```
NAME         : t                              NAME         : t-1
ABSTRACTION  : logic-design                   ABSTRACTION  : l-d-mux
INPUT        : equ-set                        INPUT        : equ-set1
SOLVING-PLAN: (t-1 (t-2 t-3 v identity)       SOLVING-PLAN: solvable
               ^ rs-mux)

NAME         : t-2                            NAME         : t-3
ABSTRACTION  : l-d-aot                        ABSTRACTION  : l-d-nand
INPUT        : equ-set2                       INPUT        : equ-set2
SOLVING-PLAN: nil                             SOLVING-PLAN: nil

Note: equ-set  = equ-set1 U equ-set2
      equ-set1 = ((p = p1 + p2) (p1 = x * -y) (p2 = -x * y))
      equ-set2 = ((z = z1 + z2 + z3) (z1 = x * y * cin)
                  (z2 = p * -cin) (z3 = -x * -y * cin))
```

Figure 4. The task frames for task t and subtasks t-1, t-2, and t-3.

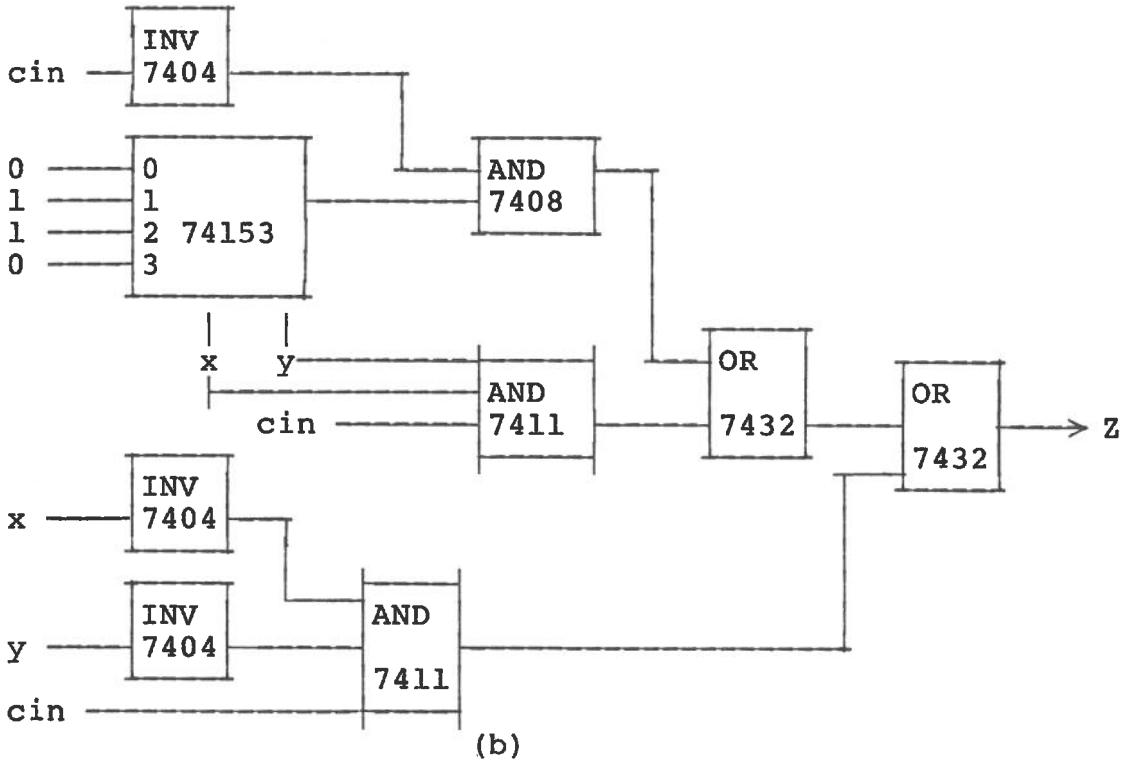
After listening to the message channel and making the necessary check, node two accepts subtask t-3. Since it is solvable, the node two initiates the necessary knowledge sources to solve it and then an implementation of t-3 is sent back to node one through the message channel.

During the next planning process, node one determines that the criterion of task t-3 is worse than that of task t-2, by comparing the actual criterion (30ns) of t-3 with the estimated criterion (15ns) of t-2, so the most promising solution plan is revised to become (t-1 t-2 ^ rs-mux). Again, subtask t-2 is broadcast.



Note:
problem solving plan1 --- (t-1 (t-2 t-3 v identity) ^ rs-mux).
problem solving plan3 --- ((t-2-1 t-2-2 v identity) ^ rs-aot).

(a)



(b)

Figure 5. (a) The message flow diagram of the logic design example. (b) The final implementation.

After making the necessary check, a problem decomposition process is initiated at node three. A problem solving plan is created as ((t-2-1 t-2-2 v identity) ^ rs-aot). The task frames of subtasks t-2-1 and t-2-2 are the same as that of task t, except the values of slots are different. According to the criteria stored in its meta-knowledge database (12 ns for t-2-1 and 25ns for t-2-2), the most promising solution plan is (t-2-1 ^ rs-aot). The subtask t-2-1 is an unsolved task, so it is broadcast. Since t-2-1 is a solvable task to node two, an implementation of it is received from node two.

After receiving the implementation of subtask t-2-1 from node two, node three learns its own expertise is better than others, so it then applies the necessary knowledge sources to solve it (t-2-2). An implementation is then formed for task t-2 and sent back to node one.

Once node one receives the result of task t-2, an overall implementation is synthesized from the best implementations of its subtasks. This implementation is then returned to the user as an answer. A messge flow diagram and overall implementation are shown in figure 5.

CONCLUSIONS

A node organization designed to support a distributed problem solving approach is presented. Among the major components of the node organization are a dynamic planning ability which guides the problem solving process in the most promising direction, a question-and-answer mechanism with meta-knowledge to control the quantity and the direction of message flow through a message channel, and a database of meta-knowledge about the areas of a node's own expertise and estimates of the abilities of other processing nodes. These features allow the system to constantly improve its performance, and to perform dynamic planning and intelligent internode communications. These features of the system are successfully demonstrated by results from a logic design example.

REFERENCES

[1] R. G. Smith and R. Davis, "Framework for Cooperation in Distributed Problem Solving," IEEE Trans. Systems, Man, Cybernetics, Vol. SMC-11, No. 1, pp. 61-70, January 1981.

[2] V. Lesser and D. Corkill, "Functionally Accurate, Cooperative Distributed Systems," IEEE Trans. Systems, Man, Cybernetics, Vol. SMC-11, No. 1, pp. 81-96, January 1981.

[3] D. McArthur, R. Steeb, and S. Cammarata, "A Framework for Distributed Problem Solving," Proc. National Conf. on Artificial Intelligence, pp. 181-184, August 1982.

[4] V. Lesser, et al., "A High-level Simulation Testbed for Cooperative Distributed Problem Solving," Proc. Third International Conf. on Distributed Computing Systems, Miami, FL, pp. 341-350, October 1982.

[5] N. Nilsson, Principles of Artificial Intelligence, Tioga Publ. Co., Palo Alto, CA, 1980.

[6] L. D. Erman and V. R. Lesser, "A Multi-level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge," Proc. Fourth International Joint Conf. on Artificial Intelligence, USSR, pp. 483-490, September 1975.

[7] J. Pearl, A. Leal and J. Saleh, "GODDESS: A Goal-directed Decision Structuring System," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 3, pp. 250-262, May 1982.

[8] M. N. Huhns, L. M. Stephens, and J. D. Yang, "A Structure for Distributed Expert Systems Based on Dynamic Planning," submitted to IJCAI-83, Karlsruhe, West Germany, August 1983.

[9] M. N. Huhns, L. M. Stephens, and R. D. Bonnell, "Control and Cooperation in Distributed Expert Systems," Proceedings of IEEE Southeastcon, April 1983.