AN EXPERT SYSTEM FOR PLANNING AND SCHEDULING IN A TELEROBOTIC ENVIRONMENT (CONTRACT #: NAG-1-763)

[illegible]

26533

P-121

Final Report

Submitted To: Automation Research Branch
NASA-Langley Research Center
Hampton, VA 23665-5225

Funding Period: May 18, 1987 - February 28, 1991

Submitted By: Celestine A. Ntuen, Ph.D.
Eui H. Park, Ph.D.
Department of Industrial Engineering
North Carolina A&T State University
Greensboro, NC 27411

June 1991

AN EXPERT SYSTEM FOR PLANNING AND SCHEDULING IN A TELEROBOTIC ENVIRONMENT (CONTRACT #: NAG-1-763)

Final Report

Submitted To:	**Automation Research Branch** **NASA-Langley Research Center** **Hampton, VA 23665-5225**
Funding Period:	**May 18, 1987 - February 28, 1991**
Submitted By:	**Celestine A. Ntuen, Ph.D.** **Eui H. Park, Ph.D.** **Department of Industrial Engineering** **North Carolina A&T State University** **Greensboro, NC 27411**

June 1991

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EXHIBITS

LIST OF TABLES

ABSTRACT

Planning and scheduling in a telerobot system are tasks which are complex because of the human-machine requirements that must be addressed. In a human system, these issues can well be approached from behavioral models. On the other hand, robot systems can be planned algorithmically by exploiting the available computational techniques.

A teleoperation, i.e; the use of telerobots for remote operations requires direct cooperation between the agents involved in the system. Thus, a methodology to achieve such cooperation must be developed. In this project, we have developed a knowledge based approach to assigning tasks to multi-agents working cooperatively in jobs that require a telerobot in the loop. The generality of our approach allows for such a concept to be applied in a non-teleoperational domain.

Our planning architecture known as TOP (an acronym for **T**ask **O**riented **P**lanner) uses the principle of flow mechanism and the concept of planning by deliberation to preserve and use knowledge about a particular task. The TOP is an open ended architecture developed with a NEXPERT^{TM} expert system shell and its knowledge organization allows for indirect consultation at various levels of task abstraction.

Considering the fact that a telerobot operates in a hostile and non-structured environment, task scheduling should respond to environmental changes. We have developed a general heuristic for scheduling jobs with the TOP system. Our technique is not to optimize a given scheduling criterion as in classical job and/or flow shop problems. For a teleoperation job schedule, criteria are situation dependent. A criterion selection is fuzzily embedded in the task-skill matrix computation. However, we have emphasized goal achievement with minimum expected risk to the human operator.

CHAPTER I

INTRODUCTION

1.1 Problem Environment

A telerobotic system consists of the use of robots or general manipulators, and humans for remote operations. Such operations are generally referred to as teleoperations [61].

A teleoperated work environment is an example of a human-machine system. Thus, for such an environment to be useful for what it is intended for, the following characteristics must be available:

1) The system operators or agents must cooperate symbiotically, at least at the highest level of abstraction [6,54].

2) The system must acquire an explicit mode of communication. An explicit communication mode is dialogue based which can provide direct interaction through devices such as a joystick, mouse, visual displays, voice synthesizers, etc. [63,69].

3) The teleoperator must have sensors and actuators, perform useful work on its environment, and be controlled remotely by other operators [10,15,21].

These three general characteristics are conceptualized by Sheridan [60] as shown in Fig. 1. Ntuen and Park [44] have also presented a general architecture which describes the environment at two levels: functional and operational levels [See Fig. 2].

At the functional level, the telerobotic system requires: a) a control model for execution of tasks; b) a supervisory model for issuing directives; c) a monitoring model for diagnosis and maintenance of system operations; d) a planning model for managing constraints and scheduling tasks; and e) a distributed problem solving model for communication between models.

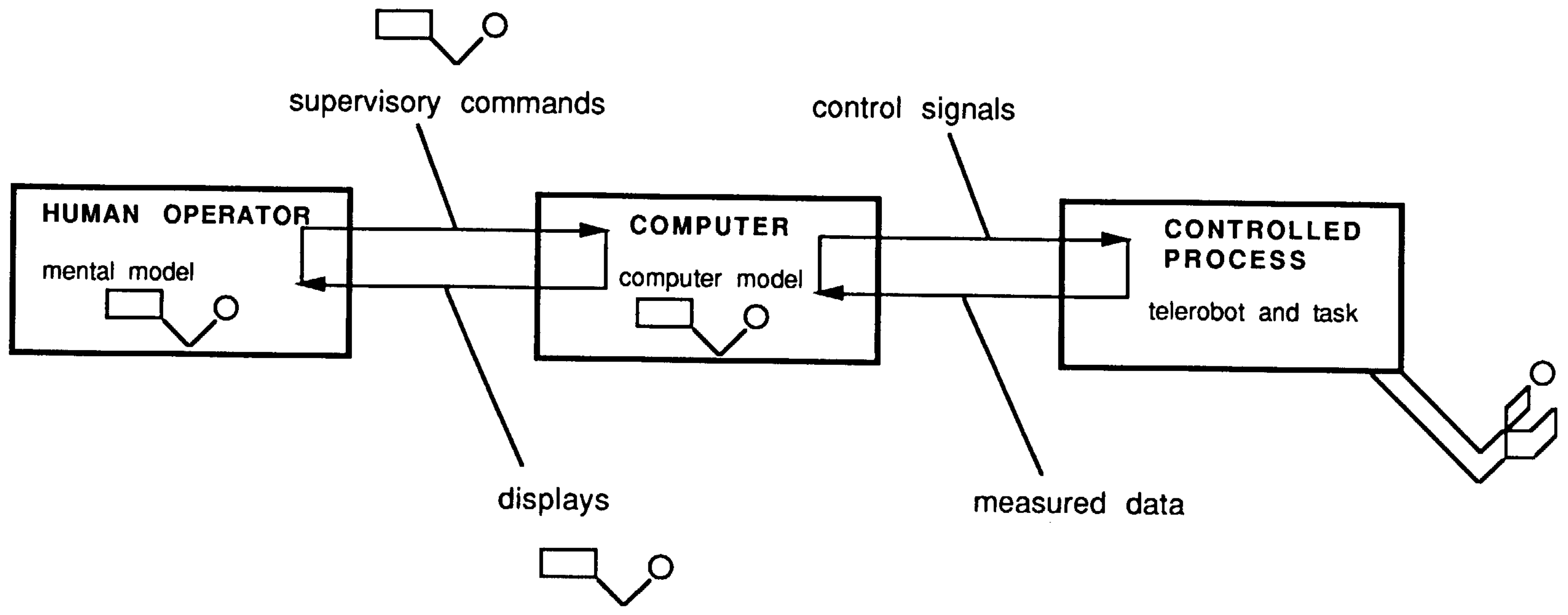


Fig. 1: A Conceptualization of The Telerobotic Environment (Adapted from Sheridan: Models of Controlled Process Internal to Human, Computer, Command Language and Display, pp. 32, 1988)

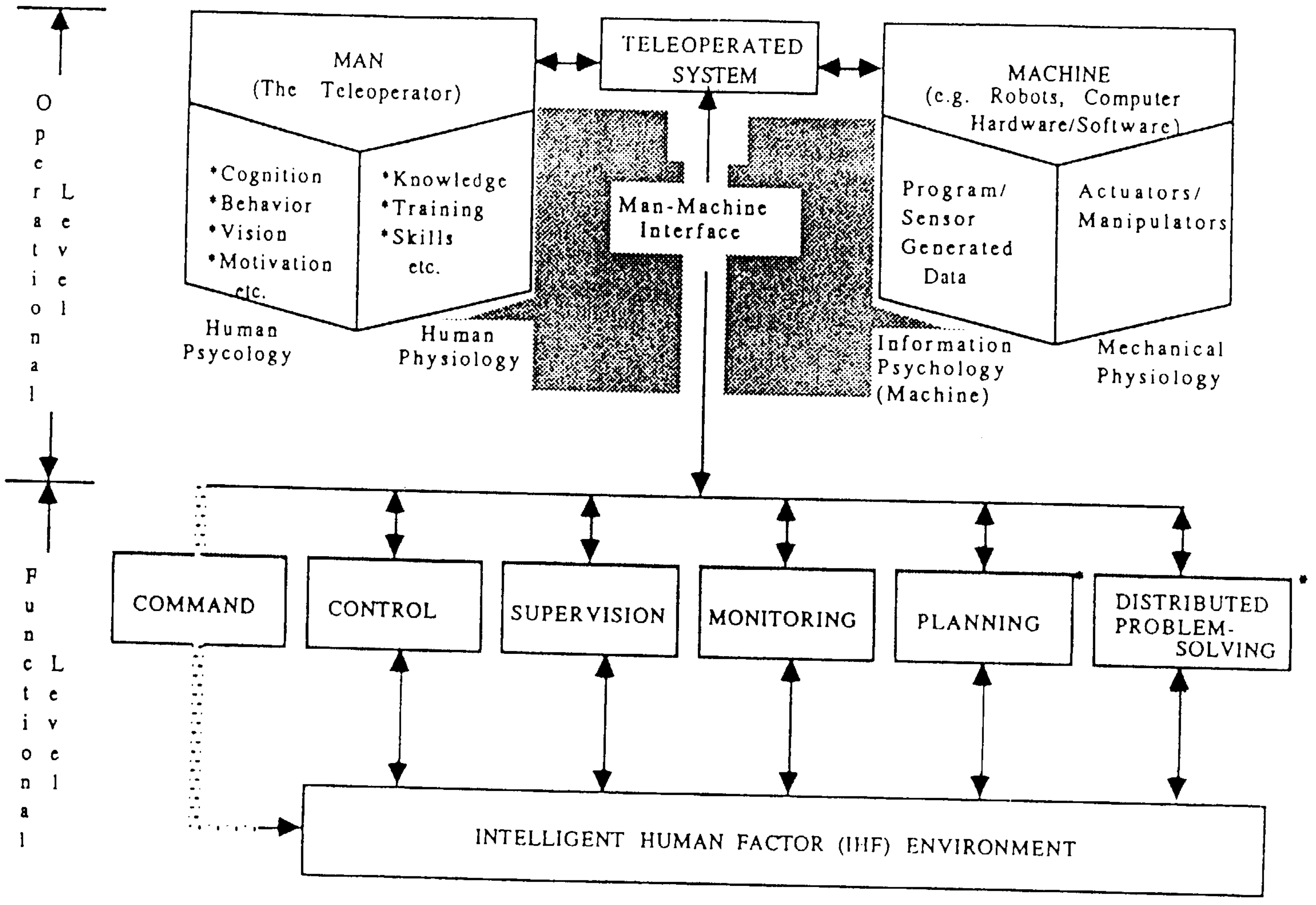


FIG. 2: Elements of a Teleoperated System.
(NOTE: * denotes advanced features)

The operational level is the highest level of the teleoperated system. At this level, mental models can be used to represent task scripts abstractly. The issue of the roles the human(s) and machine(s) should play in the symbiont system is addressed conceptually.

As described above, classical modeling tools used in robot control and manipulation fall short of addressing a telerobotic environment. This is so since some degree of human operations is required, and the robots act as aids to the operator. Coiffet [8] listed the possible areas of modeling complexities as follows:

1. The acquisition and presentation of relevant and easily interpretable information to the operator. Examples of this include the presentation of the stereoscopic view of a gripper, or the indication of the forces existing between two components that are to be fitted together in an assembly process.

2. The automatic monitoring of an operator's movements and the provisions of starting signals which activate the interruption of transmission from master to slave when the precision of the operator is failing. This function is also concerned with a system's self-testing facilities.

3. The automation of various functions so freeing the operator. These might include the gripping of objects when a device is in an automatic mode, or the maintenance of the horizontal when grippers are used to handle fluids, irrespective of the motion associated with their handling.

Since all three problems cannot be solved in a single model, we have chosen to address problems 1 and 3. Our approach is to utilize rule-based expert system technologies for planning and scheduling in a telerobot environment. We recognize that using an expert system will compensate for the most difficult problems associated with the mathematics of robotic control [See, e.g., 3,4,17]. Our expert system is generic and allows for generalizations in the world of domain-specific planning. This report further highlights the development of dynamic planner by incorporating Monte Carlo simulation techniques into situations with uncertainties.

1.2 Outline of Report

This project report is organized as follows:

- Chapter 2 presents a literature survey on planning and scheduling techniques for possible technology transfer into telerobotic applications.
- Chapter 3 presents the development of a conceptual model for planning in a telerobotic system. The concept presented is both generic and open-ended for understanding knowledge requirement in modeling teleoperation.
- Chapter 4 presents a task planning technique for a telerobotic environment. The methodology lies primarily on task consideration and resource needs in planning a multi-agent system.
- Chapter 5 presents a model for scheduling resources in a telerobotic system. A heuristic algorithm that employs results from the planning is discussed.
- Chapter 6 presents some sample applications in aircraft turnaround functions and simple block-world assembly problems respectively.
- Chapter 7 concludes the project report with a summary of what has been accomplished and a discussion of the directions for future work in planning multiagent systems with human-machine interaction in mind.

CHAPTER 2

LITERATURE SURVEY ON PLANNING AND SCHEDULING TECHNIQUES FOR POSSIBLE TECHNOLOGY TRANSFER INTO TELEROBOTICS APPLICATION

2.1 A Review of Work In Telerobotics

The framework of research works in telerobotic systems can be summarized from Sheridan's [60] metaphorical statement:

> "A telerobot is a teleoperator which also embodies understanding, memory, and decision capability so that the human operator, as a supervisor, may communicate to its high-level goals and contingencies and receive high-level state information, while the machine executes low level functions and pieces of the task semi-autonomously by closing the loop through its own effectors, sensors and internal computer."

The above observation presents the stratification in research methodologies for telerobotic environments. In summary, research in telerobotics has evolved in at least five directions as follows:

2.1.1 Control

This involves changing the teleoperator actions according to some emerging plans. A survey in this direction can be seen in Antsaklis [3], Martin & Kuban [37], Saridis [57], Yang et al [72], and Norcross [42].

2.1.2 Supervision

A supervisory control system is basically a feedback system with the capability to monitor the actual operating state of the system and to keep it within the specified target domain [13,15,23], to coordinate disjunctive efforts [7], to supervise learning such as using a joystick to train the robot [15,22,26,62], and to manage strategic decisions such as giving directives and overriding policies or priorities [2,24,43,44].

2.1.3 Distributed Problem Solving

Distributed problem solving is an issue currently being addressed in telerobotic system research. Distributed problem solving is concerned with hierarchical and parallel problem solving at the system level using global models of abstractions [10,21,34,49]. The idea is that if several computers or teleoperators can be delegated to do tasks which they can do best, then a significant amount of problem solving time can be realized. Fundamental works in this area are discussed by Koivo and Bekey [31], Silverman [63], Smith and Davis [64].

2.1.4 Monitoring

Monitoring a symbiont system is more difficult than monitoring an ordinary single-agent system. This can be explained from the fact that each agent has behavior which may be significantly different from these of the other agents or the overall system's goal(s) and intention(s). Research suggestions and directions in system monitoring have focused primarily in the area of real-time observation [29], fault diagnosis and inspection [24,54], parameter evaluation and estimation [11,32,36], and performance auditing [35,41].

2.1.5 Communication

Communication research in a human-machine system seems to emphasize a mixture of implicit and explicit models of communication.

Explicit communication is a dialogue-based communication that requires the human to communicate with the task allocator using an input device such as a keyboard, mouse, or lightpen, or by using his voice, buttons, or switches. Although this type of communication has the advantage of minimizing misunderstanding in intent between the human and the task allocator, it is unfortunately costly in terms of taking up more of the human's time due to the human having to stop performing tasks to communicate with the task allocator [24,48].

Implicit communication is a model-based communication in which the computer uses models of the human to predict what the human is likely to do next. From this prediction, the computer attends to tasks which are likely to be neglected by the human. Implicit communication is typically used when the human performs the majority of the tasks. This type of communication has the advantage of allowing the human to execute tasks without having to communicate with the task allocator. The disadvantage of this method is that it requires the development of an appropriate predictive model of human task-selection performance which is usually difficult to build and results in an imperfect model of the human.

2.2 Prototype Systems Developed For Telerobotic Applications

Several test-bed systems have evolved since the 1980's which are dedicated to teleoperated functions. Among these are the following:

2.2.1 TRSS - Teleoperator/Robotic System Simulation

TRSS [26,48,49] is a modular software simulation coupled to a reconfigurable teleoperator control station. The module resides in a relatively powerful processor that can coordinate communication from other processors and devices. The module deals with "strategic" task planning, database management of the machine's concept of the environment, supervisory monitoring of the teleoperator control station, and the interfaces to various "tactical" controllers.

2.2.2 DAISIE - Distributed Artificially Intelligent System for Interfacing with the Environment [49]

This is implemented at NASA Langley as a system that divides the control of a sensor/cognition/actuator system into two major hierarchical levels. These levels are termed strategic and tactical. The "tactical" level refers to local, specific control of a particular sensor/actuator grouping (preceptor/proprioceptive actuator). "Strategic" refers to a control level with a global view of all tactile units and their actions. DAISIE implements the concept of using various degrees of abstraction at different goal levels.

The DAISIE system exploits distributed processing within the limitations of the available hardware. Functions such as manipulation, vision, end-effector control, and force-torque sensing are each run by separate processors. The higher "intelligent" levels are also distributed in separate processors.

2.2.3 TART (Teleoperator and Robotic Testbed)

Harrison and Orlando [26] discussed the use of TART implemented on the VAX 11-750 in ISRL (The Intelligent System Research Laboratory). TART is a layered driven model in which each successive layer provides additional value to the system. Currently, five layers are implemented: 1) user, 2) system, 3) scheduling, 4) communication and 5) servo/sensor. The lowest four layers of TART are designed for error-checking required by user applications. Users are encouraged to use only the TART-defined system level mechanisms for modification of data structures.

2.2.4 LART (Language-Aided Robotic Teleoperation Systems)

LARTS [58,59] incorporates two sets of teleoperational languages with a master-slave manipulator. Both spatial and temporal autonomy which support the operator are provided by the languages. The authors of LARTS focus on the structuredness in teleoperational task execution. For example, there may exist many constrained motions in the handling of objects. Elementary tasks, such as pick, place, remove, grasp and so on which are executed repeatedly are examples of teleoperational task execution.

2.2.5 <u>TOL.O</u> (Teleoperator-Oriented Language of the Object-Level)

To cope with the burden of the operator having to teach the actual environmental data in the program, a special teaching method designed for teleoperation shows a synopsis of the method [58,59]. TOL.O is designed to specify elementary task motions of teleoperation. TOL.O instruction yields the program for the task. The programming burden for the operator is therefore reduced. Specifications in TOL.O are as follows:

1) Operator declares aim of task using TOL.O instruction.

2) Instruction is automatically expanded into the motion-level task procedures.

3) Teaching-executing systems interprets the draft program steps one by one and prompts the operator to do the necessary motions for the task execution and teaching.

4) Operator executes the task by operating the masterslave manipulator and signals the system, using a button, that the motion is completed.

5) At the end of task execution a program consisting of motion procedures together with the environmental data is stored in the system.

2.2.6 <u>MSM</u> (Master-Slave Manipulator)

The MSM [28] language describes the software jigs and control schemes of the man-slave manipulator. The instructions related to the software-jig fall into the following three categories:

1) instruction specifying elementary motion;
2) instruction which construct the jig body by combining the constraints;
3) instructions describing the attachment and detachment of the jig body.

2.3 Planning and Scheduling Concept for Telerobotics

2.3.1 Concepts

The term "planning" has gained significant meaning in describing problem-solving protocols in the AI Community. Although the term is as old as human existence, its application in the development of problem-solving systems has given entirely different meanings to the concept.

Generally, planning methods require one to compile some qualifying information about a given problem domain. From this, pieces of information which are likely to guarantee useful contributions toward solving the problem are retrieved and organized for that purpose. We refer to this kind of planning as "naive" planning.

The artificial intelligence-based definition of planning can be described more succinctly as the systematic, vis-a-vis experimental compilation, organization, and use of domain knowledge for the purpose of automatic solution-finding to a given problem via the computer. The AI community has, in the past twenty years, been involved in the development of such systems. These are referred to by such names as "planners", "problem-solvers" or "expert system advisors."

The concept of planning for intelligent problem-solving systems can be traced to Newell, Shaw and Simon [39] in developing a planner called General Problem Solver (GPS). They introduced the "mean-ends analysis" paradigm which solves problems by applying an operator that would achieve some of the goals of the problem and take the preconditions of the operator as new goals. STRIPS [16,18], due to Fikes and Nilsson, modified the GPS concepts to that of an action model -- in which steps have post conditions that are the only things that get changed by the steps. These two planners -- GPS and STRIPS -- operated on nonconjunctive tasks in elementary "block-world" domain.

Domain-independent conjunctive planning started in 1973 with Sussman's HACKER [68], whose basic paradigm is often referred to as the "Sussman anomaly". The urge to resolve this anomaly led to a series to interests and developments in the field of planning. For example WARPLAN [67], INTERPLAN [9], NOAH [55,56], NONLIN [70], MOLGEN [66,67], DEVISER [71], SIPE [73], TWEAK [14] and ISIS [20].

The growing development in the field of planning research has led to different representational view points, constructs and implementation concerns. These concerns led to the special workshop in planning held in Santa Cruz as sponsored by DARPA [68].

2.3.2 Some Planning and Scheduling Techniques

In order to develop a planner for a telerobotic system, the current planning techniques were reviewed and summarized as follows:

1. **A Tactical Planner**: a planner which is primarily concerned with deciding what to do in situation in which available information is limited or uncertain.
2. **A Linear Planner**: In STRIPS [16] for example, actions are represented as functions from sets of sentences to sets of

sentences in some appropriate language. Such a representation allows for what is disparagingly referred to as linear planning. In a linear planning framework, plans correspond to sequences of primitive actions.

3. **A Hierarchical Planner**: This is perhaps the best known and most misused technique since a number of unclarified concepts get tangled with its application. Hierarchical planning arose out of dissatisfaction with linear planning. A hierarchical plan starts at the general level of planning abstraction and moves down to specific, details, subplans and levels. Problem-solving with hierarchical networks occur via conflict resolution [19], relaxations [33], and modeling interactions between subplans [70].
4. **An Opportunistic Planner**: This is a planning system whose actions co-routines with the model environment to dynamically instantiate or alter a problem-solving behavior based on circumstances. That is, during each point of a problem-solving life cycle, the planner's current decisions and observations suggest various opportunities for further plan development and changes. Originally suggested by Hayes-Roth and Hayes-Roth [27], the concept has been used to explore opportunistic scheduling of manufacturing systems [19].
5. **A Least Commitment Planner**: NOAH [55] and his descendants operate on the premise that operations are not to be sequenced unless absolutely necessary. A set of procedures known as critique agents are used to detect and correct interaction,

eliminate redundant operations, and so forth.

6. **Case-Based Planner**: Case-based [11,68] planning systems take as a starting premise that the organization of experience is paramount in formulating new plans and debugging old ones. Case-based reasoning is a simple idea: solve new problems by adapting solutions known to work for old problems. The Case-Based Planner offers several potential advantages over rule-based reasoning systems: rules are not combined blindly in a search for solutions.
7. **Agenda-Based Planner**: This is a class of planning system using procedural listing of objects, events, and activity occurrences during the problem-solving life cycle. Heuristic models, such as priority rankings, utility preferences, or economic values are used to "promote" or "demote" plan structures in the agenda list. An example of planner in this category is SIPE [73].
8. **Endorsement-Based Planner**: This is a planner which develops and refines plans based on the user's actions and possible intentions. The concept of "intention" allows the planner to reason from the human behavior perspective.

2.3.3 Plan Evaluation for Telerobotic Application

Successful implementation of computer-based systems are usually need-dependent. Therefore, in order to buy or develop a planning shell for a problem-solving system, several issues have to be resolved. We present the basic and most obvious technical issues:

1. **Language** -- A plan must first have a vocabulary of symbols and notations in which the initial state, goal conditions, and operators may be represented with conceptual objects. For example, the assertion: INROOM (ROBOT, ROOMA) means that the mechanical device called ROBOT is in room called ROOMA. Some languages have been developed based on the planning environment. For example, NUDGE uses FRL, CONNIVER uses MICROPLANNER, STRIPS uses MACROPS, and ISIS uses SRL.
2. **Intention** -- A plan must have a purpose or intention. This characteristic allows a planner to act in a directed, domain-specific fashion. It is possible for a plan to have multi-intentions, and subplans are developed in a layered fashion to handle such intentions.
3. **Belief System** -- A plan must contain some specification or instantiation of beliefs about the environment for which the plan is designed. The availability of an embedded belief system in a planner allows for on-line result validation and verification.
4. **Conflict Resolution Capability** -- A planner should be able to trouble-shoot the problem environment, understand and resolve basic conflicts.
5. **Cooperation** -- A planner should be able to incorporate knowledge from other (user-defined) plans towards solving a common problem. This principle is known as "Cooperative Planning" [10].

6. **Authority** -- A planner should possess an authority to determine (using its knowledge) whether to exclude one or more subplans during execution of a problem.
7. **Responsiveness** -- A planner should respond to events occurring during the plan's execution. The number and magnitude of changes may be a source of difficulty. This concept is known as "Replanning Ability" [38].
8. **Plan length and predictability** -- The more predictable the execution environment, the longer the plan can be with a reasonable expectation for successful completion. Automatic programming, in particular, can produce plans (programs) millions of steps long that usually complete successfully. Producing such big plans is only possible because a computer is such a predictable environment; the main source of unpredictability here concerns the input to the program.
9. **Correctness versus robustness** -- Traditional AI planning systems tend to concentrate on producing correct plans. Although this method is appropriate for highly predictable environments, it is much more important to produce robust plans in realistic situations. Robustness means the plan is likely to succeed no matter what unanticipated conditions arise; that is, robust plans avoid including commitments to courses to action which allow few options if they fail in execution.

2.4 SUMMARY

To summarize, AI-based planning systems can be used for the following purposes:

- Deciding on a course of action before any action is taken.
- Monitoring progress during problem-solving in order to catch errors before they create much difficulty.
- Reducing the amount of search which characterizes most conventional problem-solving systems (e.g. planning an operation schedule for a job shop).
- Providing a modular approach to problem-solving, thereby allowing for easy modification, portability, and adaptability.
- Allowing for both descriptive and prescriptive representations of how actions and human behaviors interact during problem-solving situations.
- Plans allow humans to experiment on concepts related to the design of behaviorally oriented systems.
- Plans allow humans and computers to build a pragmatic model between a hypothetical system and reality.
- Plans allow for tests and comparisons of the various ways experts solve identical problems, and recommend a common framework for standardization (where possible).
- Good plans attempt to minimize redundancies in goal and resource specifications, thereby reducing costs of problem solving.
- Plans can be used to instruct and explain concepts.

CHAPTER 3

THE DEVELOPMENT OF A CONCEPTUAL MODEL FOR PLANNING IN A TELEROBOTIC SYSTEM

3.1 Introduction

The aim of this chapter is to discuss a conceptual model required for planning a telerobotic system. As indicated in a previous study by Orlando [49], the complexity of a telerobotic system is such that "interleaving of the steps of plan creation and plan execution must be conceptualized at the highest level of abstraction prior to modeling." We show in this chapter that planning in a telerobotic system requires an understanding of a human-machine (robot) working together symbiotically. Among several other things, this can take place at four dimensions; (1) understanding the role of humans in teleoperated tasks (Expert system), (2) understanding the task environment and knowledge required to accomplish the task (Knowledge Base System), (3) understanding the communication and problem-solving approaches for a "symbiotic" system (Distributed Problem-Solving Environment), and (4) understanding the cognitive requirements that allow the agents (robots, human, computers, etc.) to learn from the accomplishment of one another (Learning System). This concept is shown in Fig. 3 and it defines the components of a telerobotic planner.

3.2 <u>The Expert System</u>: Understanding the role of Humans in Teleoperated Tasks

The use of robot technology along with human labor does not necessarily remove humans from the system in which a task is to be performed. As noted by Rasmussen... basically it moves them from

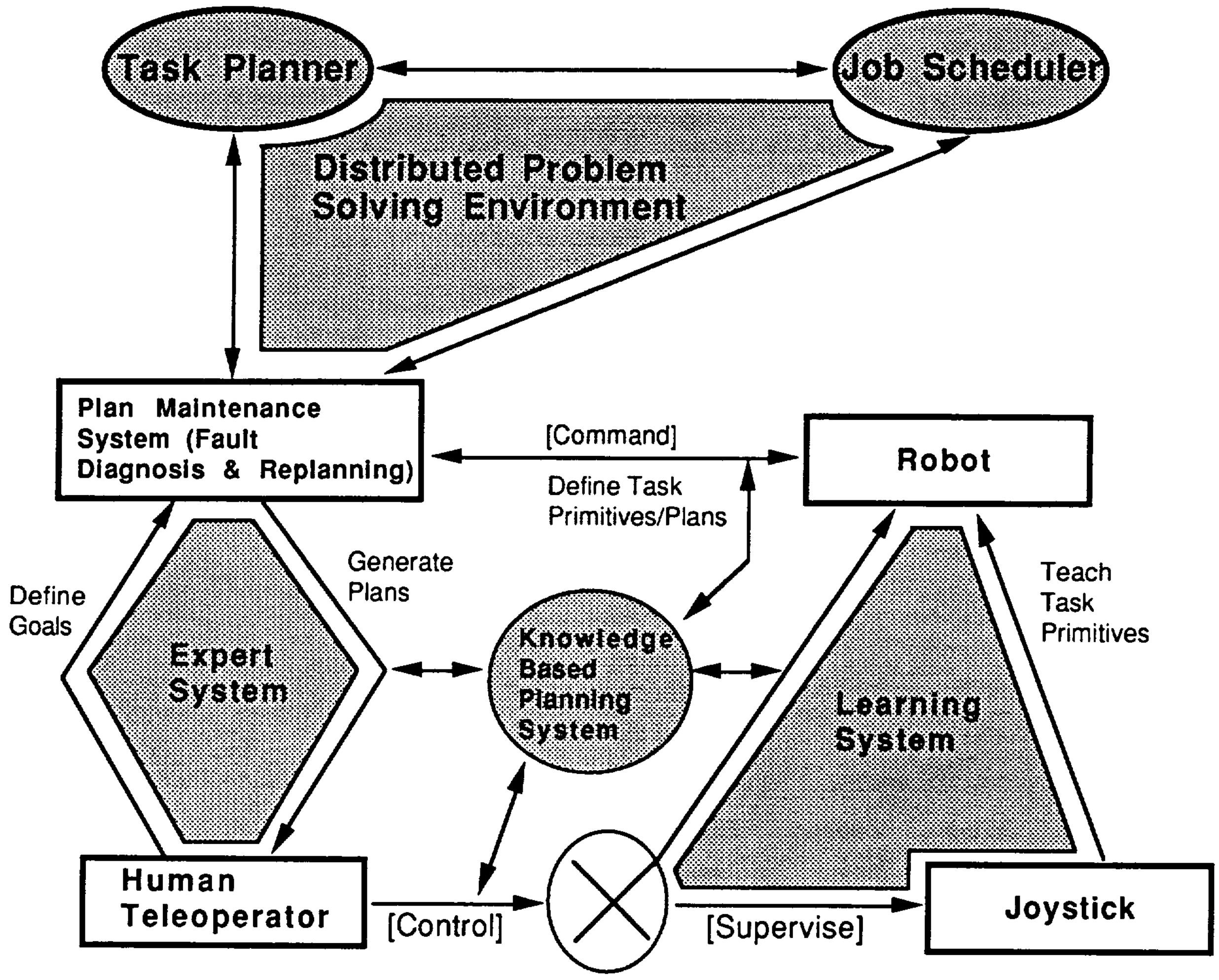


Fig. 3: A Conceptual Model Environment For Planning And Scheduling In A Telerobotic System.

the immediate control of system operation to higher-level supervisory tasks and to long-term maintenance planning tasks."

Because of the expected changes in the human role as a controller to supervisor, the control expertise must be preserved and transferred to the robot. The realization of such transfer can be achieved via expert system technologies.

Expert systems are computer-based models that can solve problems that are normally solved by the human "experts." For a telerobotic system, the roles of an expert system module can be to:

- direct both the robot and human to goal attainment;
- provide multiple context advice;
- supervise the human and the robot in task allocations;
- provide suggestions and alternatives to problem solving situations.

3.3 The Knowledge-Based System: Understanding the Task Environment and Knowledge Required To Accomplish Tasks

To solve expert-level problems in which several agents interact, an access to a substantial knowledge base is required. In fact such a knowledge base should be dynamic with response to different task environments. As identified by Ntuen, Park and Sliwa [47] there are two general types of knowledge requirements -- teleological knowledge and epistomological knowledge respectively.

Teleological knowledge relates to the entire design spectrum of the teleoperated system. The morphology of the design can be understood only through the analysis of human and machine (robotic) capabilities with respect to task requirements.

This concept is referred to as "mixed initiative" [48]. "The mixed initiative concept is based on the transfer of authority and control responsibility between an automated system and a human operator." Thus, the knowledge (information) for planning relates to task, control, and feedback.

Examples of teleological knowledge are robot control algorithms, human discrimination of occluded environment, or cognitive skills required for a particular task (domain knowledge).

Epistomological knowledge relates to the information and data required for understanding a task situation defined at the highest level of abstraction. An introduction to epistomological constructs for teleoperated systems has been discussed by Orlando [47]. In her discussion, an evolutionary approach where knowledge is activated, complied, managed, and controlled from the bottom-up is presented. The knowledge elements identified for modeling teleoperations are as follows:

1. Intrinsic and extrinsic data compilation. This involves a process of deep generation of information to describe a system behavior. Psychologists refer to this as a pseudo bio-feedback information processing.[1]
2. Thematic and rhetorical knowledge of the system. This involves some sense of spatial cognition where information about a situation is stored in the form of dynamic production

[1]Rasmussen, J. (1983). "Models of Mental Strategies In Process Plant Diagnosis". In Human Detection and Diagnosis of System Failure (M.J. Rasmussen & W.B. Rouse, Ed., Plennum Press, New York).

rules.[2]

3. Introspective Knowledge. This involves data/information based on reflective assimilation of the environment. In the telerobotic system, a simulated perceptual representation of a picture by computer vision can provide informal introspective data.

4. Perceptive Knowledge. This involves data based on physical sensation as interpreted in the light of experience. Orlando [48] used a theory of physiological psychology to explain the "hierarchy of abstraction" of knowledge evolution. Perceptive knowledge for robotic software systems requires instantaneous cognition of "foreign" objects that may pose a threat to mission participants.

5. Catalog of Intentions and Meanings about Concepts. These relate to data used to describe the purpose of the (planning) system and expected goals. The data or constructs can be ill-structured, fuzzily described and/or possess temporal attributes.

3.4 Distributed Problem-Solving Environment

A telerobotic system requires that humans and artificial agents (telerobots and computers) cooperate in decision-making and control of tasks in a complex, unstructured, and dynamic

[2]Manfred, K. and Galanter, E.H., 1958. "The Acquisition and Utilization of Information in Problem Solving," Information and Control, Vol. 1, pp. 267-288.

environment. Unlike the manufacturing robots, telerobots are expected to possess "exceptional" intelligence, be flexible, and exhibit a high level of dexterity and reliability. The existence of these skills and requirements represent "virtual" attributes which must match those of the human operator -- at least conceptually if the true meaning of "symbiosis" is to be modeled into cooperative and distributive planning. Interests in telerobotics require work on cooperative problem solving. In this environment, we conceive of a group of teleoperators who form a "crew" to perform a particular task known as action units. Many modeling issues of concern for effective cooperative problem solving include:

(1) Agents should co-routine during task execution. Thus, task assignment based on sub-task decomposition should rely implicitly on the agent's capability.

(2) A distributed (and cooperative) problem solving environment should possess at least one agent who can "authorize" the execution of plans. Thus, the idea of priority is crucial in such an environment.

(3) Agents need to represent and reason about their own actions, the actions of other agents, and their interactions between agents.

(4) Agents need to coordinate their interactions and possess interleaving plans so that they work as a coherent team when cooperating to achieve a shared goal.

3.5 The Learning System: Understanding The Cognitive Requirements In a Telerobotic System.

The capabilities of "symbiotic" system agents to learn from one another is an issue of concern in modeling a telerobotic environment. There is significant support in the literature by cognitive theorists [4,39] that suggests the need of learning as a paradigm in which a human-machine environment can work cooperatively. Because many tasks are difficult to perform jointly in a telerobotic system, especially if communication time delays exist, an appropriate learning mechanism is needed. Such a learning mechanism may involve human intelligence (knowledge), sensory and robotic capability to perform remote manipulation tasks, etc.. A particular consideration of learning in a telerobotic system is skill acquisition by robots.

Skilled tasks are multi-componential and heterogenous in nature, requiring mixtures of cognitive, motor and perceptual abilities [1]. Acquisition through the learning of performance strategies is typically necessary when tele-autonomous systems are desired. Thus, for a telerobotic application, a learning system should interact between the human teleoperator and the robot such that:

(1) The human can teach basic task primitives to the robot.

(2) The robot can learn what is being taught through cognitively driven models.

(3) The robot can learn about task environments, generate plans, and respond to schedule or scenario changes.

3.6 SUMMARY

A teleoperation requires that both human and artificial agents (robots) cooperate in decision making during task execution. Task planning in such a system requires more than the classical modeling techniques because of interactions of several non-quantifiable parameters. One approach to modeling is therefore first to conceptualize the levels of abstraction, and then construct a domain-specific simulation of the task environment. We have identified and discussed these levels of abstractions, the roles of human (experts), the knowledge base requirement, the distributed method of problem solving that integrates the human and machine (robot) capabilities, and the need for skill acquisition model between agents using the concepts from learning literature.

CHAPTER 4

TASK PLANNING IN A TELEROBOTIC DOMAIN

4.1 Overview of the Planning Method Used

Task planning is one of the several issues in a telerobotic system, as well as in artificial intelligence. The main concern is if a telerobotic system is considered to be "symbiotic", then the interdependence of its agents comes from sharing the same limited resources. Each of the agents has its own state control variable, behaviors, and goal functions. Thus, the resources of the team of agents must be efficiently mapped onto their capabilities and the demands of the prevailing task regardless of its changing nature and uncertainties.

Planning problems, like most AI topics, have been attacked in two major ways [53]: approaches that try to understand and solve the general problem without the use of domain-specific knowledge and approaches that directly use domain heuristics. In planning, these approaches are often referred to as domain dependent (those that use domain-specific heuristics to control the planner's operation) and domain independent (those in which planning representation and algorithms are expected to work for a reasonably large variety of application domains).

The planning and allocation of tasks in a telerobotic system environment are rather complex. In a single unit system, the behavior of the entities can be simulated as a "snapshot" of a defined scenario since the entities have predefined roles to play. A telerobotic system on the other hand is a symbiotic system which requires division of work between the humans and the machines in such as a manner as to facilitate their cooperation through shared knowledge, skills, and experiences. This usually involves multi-tasks and multi-agents (humans, computers and robots).

The multi-task problem is a collection of several tasks operating concurrently as opposed to only one task operating at a time (sequential task problem). The task allocation strategy may use a static model, i.e., assign a fixed sub-set of the tasks to each resource prior to job execution. This type of allocation is predominant in activity-based simulation environments where the resources perform only their tasks when necessary. The basic problem in this type of allocation is that if one resource fails in performing its task, another resource cannot take over the operation of the task. A more flexible assignment strategy is the dynamic approach. In dynamic task assignment, any resource which is currently free and capable to perform a task could be assigned the next task to be performed. Typically, the planning token would normally consist of:

- Planning and allocating tasks for humans and machines.
- Planning resources to meet the desired goal(s).
- Sequencing tasks according to demand (e.g. control, command, supervision and monitoring.

Harrison and Orlando [26] give the following conceptualization for telerobotic task planning:

1. Planner - system plans and executes operations in respond to task request and with fixed scenarios. The system appeals to operator when anomalies arise.

2. Expert planner -- same as planner except that plans are synthesized from generic subtasks and alternates are logically derived.

3. Robust planner -- same as expert planner except that alternates are developed dynamically in response to feedback and performance.

4. Learning planner -- the system is able to autonomously improve its performance by generalizing circumstances of previous tasks.

In relation to the definition and classification of planning technologies discussed above, at least two kinds of planning can be envisaged for a telerobotic system: path planning and structured planning. Path planning occurs at the spatial level where task plans are geared towards changes in position orientation in time and space. The major research in this area is in collision avoidance. The spatial planner determines how to transfer objects and move through a work space without collision with obstacles.

Structured planning deals with temporal task procedures which include such elemental tasks as pick, place, remove, inspect, and so on. Usually, these task elements are executed repeatedly in a teleoperated mode even in an unstructured environment. The task-level planner specifies the activities for each component of the work space in terms of sequencing for efficient operation.

4.2 Plan Formalism For a Telerobotic Domain

The following definitions will be used throughout this chapter:

X	a task environment
P	a plan primitive
G	a vector of goal states to be achieved (g x 1 in dimension)

S a vector of subgoals where the element s_{ij} is the subgoal j required for goal achievement g

$p = 1, 2, \ldots . P$
$g = 2, 2, \ldots . G$
$j = 1, 2, \ldots . S$

θ is a matrix of operator elements with pxj dimension. Thus, the operator elements θ_{pj} is the transformation or problem solving operator required to move subgoal j to goal state g.

A Means-ends scores for plan strategy p; a vector px1 in dimension

The plan formalism can be written as

$$P(G):\theta * S \rightarrow I * GA^{T} \ldots\ldots\ldots\ldots (1)$$

Equation (1) can be stated as follows: in order to achieve goal G, use plan P which transforms a vector of useful subgoal S to goal state G using the operator θ. I is g * g identity matrix.

By using the method of means-ends-analysis [39] and Noah's [56] least commitment approach, we view a telerobotic task planner evolving in a dynamic environment of cooperating behaviors. Thus, agents can recognize unnecessary plans and eliminate them. Similarly, new opportunities can be learned and new plans added. In this case, the planner changes with respect to various subgoals, and the operators must learn a new syntax from every new task environment. Example task scenarios with a new syntax can be the use of a telerobot for burial of waste material, or assembly of space structure. As can be envisaged, the task environments are different; however, there are certain task primitives such as material handling that are common. Included this concept, a domain-specific plan with "syntactically scoped variable" with constraint is written as:

$$\forall_n \exists_x \left\{\frac{d\theta}{dx}\right\} \frac{dA}{d^n S} = P*(G) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\text{where } n = 1,2\ldots g,$$

$\frac{d\theta}{dx}$ is the dynamic changes in operator (problem solving strategy) for new problem scenario x, $\frac{dA}{d^n s}$ represents changes in means-ends scores with respect to new subgoals; $P^*(G)$ is the existence of optimal plan required for G.

Similar to STRIP [15] and ABSTRIPS [17], the sequence of actions:

$$ADD(\theta) \rightarrow DELETE(\theta) \rightarrow \ldots\ldots$$

can be written abstractly as

$$\theta(x) \rightarrow (1 + Sign\ A) \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

where $\theta(x)$ is the operator required for problem environment x and

$$Sign\ A = \left\{\begin{array}{l} 1\ ,\ \text{if } ADD(\theta) \text{ is true} \\ 0,\ \text{if current operator is true for } x. \\ -1,\ \text{if } DELETE(\theta) \text{ is true} \end{array}\right\}$$

4.3 TOP: The Task Oriented Planner For Telerobotic Application

4.3.1 TOP Concept

The concepts of plan formalism discussed have been used to develop a prototype planning environment known as TOP (a Task Oriented Planner). TOP is a planner that uses information from a task environment to plan and schedule resources towards the achievement of a goal. The data structure for TOP is at three macro levels which are:

1. <u>Goal Criteria.</u> The user explicitly specifies the task (goal) to be accomplished and the relevant criteria for effectiveness. For example, a goal can be aircraft refueling, and the measure of effectiveness in this case could be the completion (turnaround) time.

2. <u>Resources.</u> Each task domain needs resources for task execution. In the telerobotic system, the human telerobotic interaction requires a different modeling procedure for task assignment. Thus, for each resource, one has to specify the skills required to accomplish the desired task. The skill level can be vision, search and knowledge. For each skill, a fuzzy membership function is used as a trade-off decision model at the task assignment phase (this is discussed in detail in section 4.4.1 of this report).

3. <u>Tasks.</u> At the lowest level of abstraction, a task is used in the same context as a job. In the highest level abstraction, a task represents some chunks of jobs with some defined relationships. The development of a task tree is a required step for the TOP (See Chapter 5 for further discussion).

4.3.2 TOP Structure

The overall TOP concept including the planner is shown in Fig. 4. As shown in Fig. 4 task planning is accomplished by the PLANNER. The planning concept extends the methods of the GPS means-ends analysis and NOAH's least commitment approach to include deliberation. The "acts of deliberation" take into consideration the possible time lags between the agent's behavior during task planning phase. Thus, a deliberate plan is driven by possible expected behavior of the agents taking part in the task domain.

In order to minimize the time lag at the planning phase, we have developed a tool known as a "Deliberate Plan Network" (DPN). DPN uses the principle of flow mechanism to preserve certain

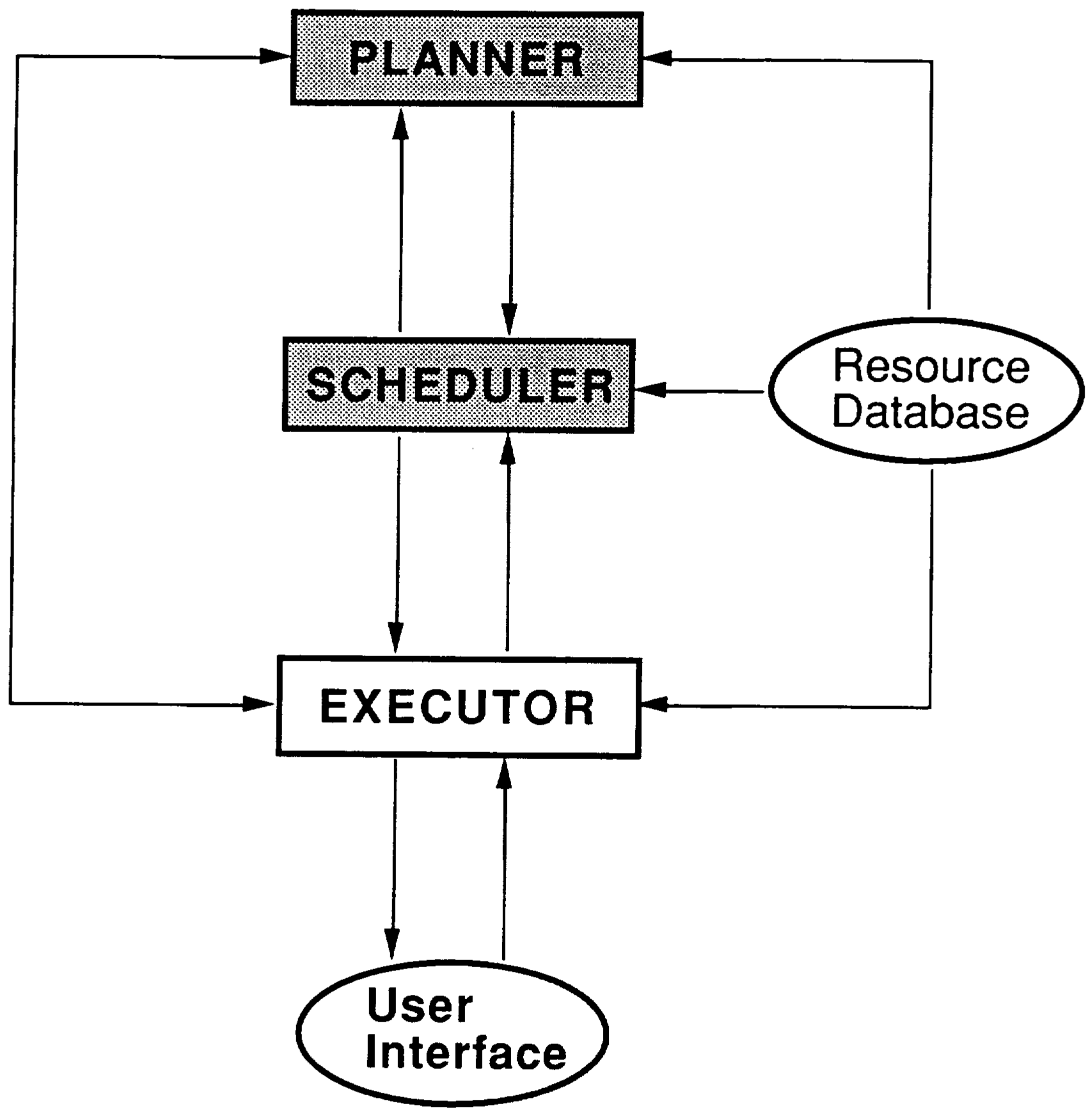


Fig. 4 : TOP Architecture.

logical primitives from being considered repetitively. In so doing, we try to minimize assigning tasks to agents at more than their desired capability.

In the DPN, we have tokens of decision elements known as bureaucratic fixers (BF) and moderators. A BF tries to evaluate the agent's capability for each task and flags out an agent dominating another during task allocation. The moderator is similar to constraint resolution criterion that controls the replanning and recursive behavior inherent in DPN. Appendix-A gives an abridged discussion on DPN.

4.4 Task Assignment in TOP

The tasks performed in a real-world environment are typically characterized as requiring "skilled performance", and may require integration of multiple types of skills (motor, perceptual, procedural, etc.)

The scenario above can best be described by a mix of all or some of the following:

- dynamic in that changes of states occur;
- real time in that decisions must be made instantaneously;
- unpredictable environment in that rules and behaviors describing the task environment are not stable;
- multiagents are involved in that decision policies, plans, etc., coexist in a spatially distributed fashion.

4.4.1 The Teleoperator Skill Matrix (TSM)

In order to develop a multi-task assignment model, the concept of teleoperator skill matrix (TSM) was formulated. A TSM is a multi-dimensional matrix that contains weighted information on a task to be performed and the operator skills required to perform such a task. The TSM is developed in stages as follows:

Stage 1: Identify the tasks to be performed and the resources required to perform the task (See Exhibit 1).

Stage 2: Identify the skill inventory for the resources (See Exhibit 2). An example skill inventory list (See, e.g.; Nof, Knight & Salveny [41]) is given below:

1. Vision
2. Manipulation
3. Search
4. Recognition
5. Knowledge
6. Capability:
   - payload
   - computational
7. Motion
   - dexterity
   - maneuverability
8. Reasoning
9. Communication
10. Endurance
    - tolerance of adverse environment

Stage 3: Subjective rating of the resource skill using the method of Parker and Pin [50]. The rating score is defined as a fuzzy membership function.

Exhibit 3 shows a multidimensional TSM which is a combination of Exhibits 1 and 2. The matrix entry and solution to task assignment is discussed below. Note that each subtask is rated based on the resource profile. A typical task tree is shown in Fig. 5.

4.4.2 Task Assignment Model Using the Fuzzy TSM

The concept of fuzzy set introduced by Zadeh [74] describes a situation in which the imprecision is due to vagueness or subjective judgement rather than randomness.

With this aim in mind let us define a fuzzy set as described by Zadeh. A fuzzy set is a class of objects with a continuum of

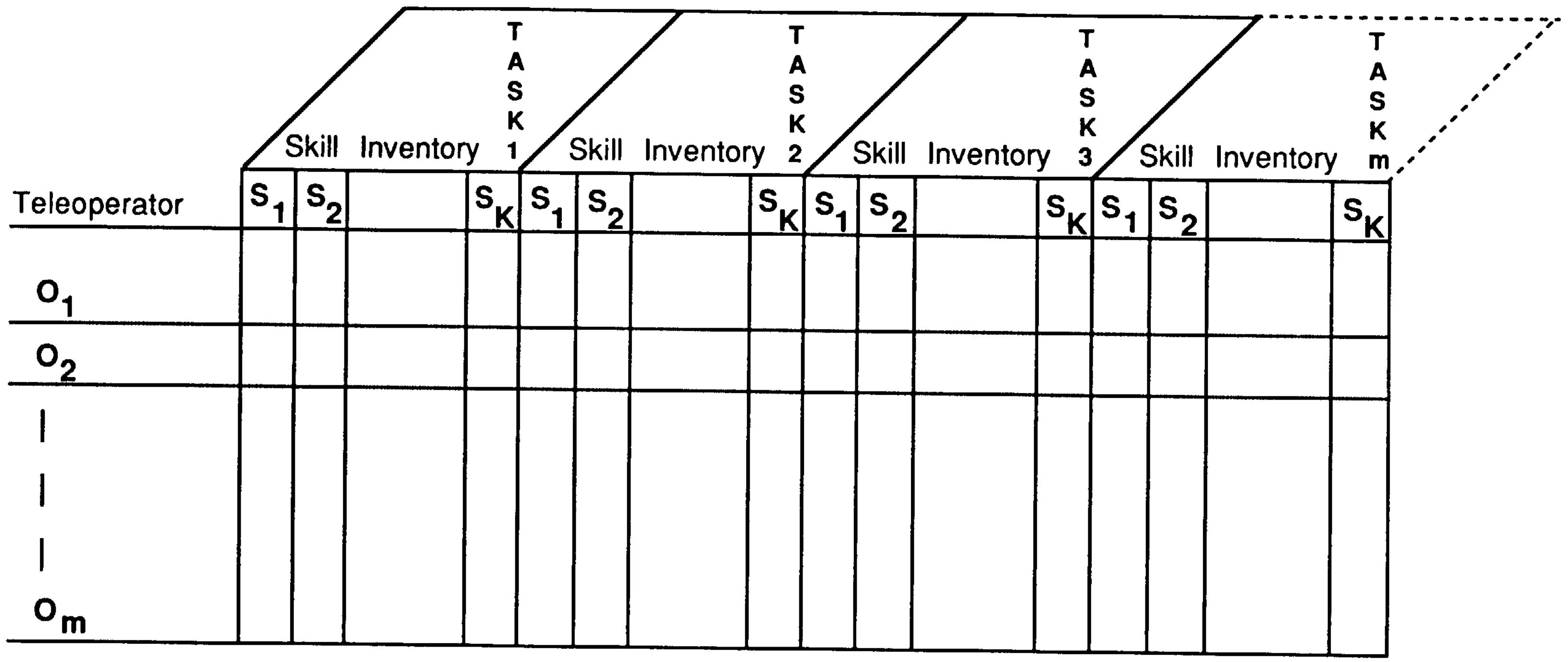
Resources	$Task_1$	$Task_2$	...	$Task_m$
R_1				
R_2				
. .				
R_n				

Exhibit-1: Task - Resource Matrix

Resources	S_1	S_2	...	S_k
R_1				
R_2				
. .				
R_n				

Exhibit-2: Resource - Skill Inventory Matrix

Exihibit 3: TELEOPERATOR SKILL MATRIX (TSM)



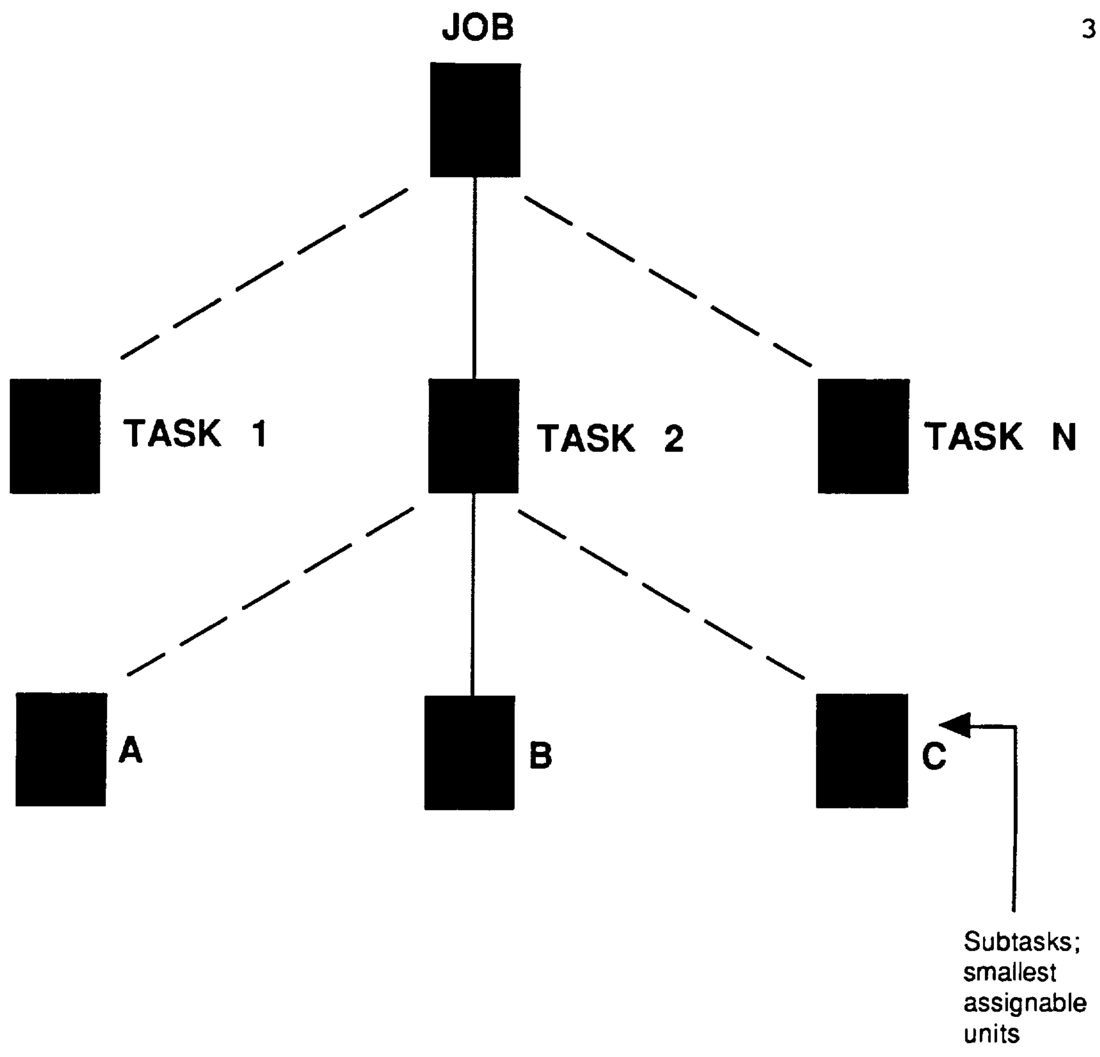


Fig. 5: An Example Task Tree.

grades of membership. Such a set is characterized by a membership (or characteristic) function which assigns to each object a grade of membership ranging from zero to one. Let X be a space of point objects, with a generic element of X denoted by x. A fuzzy set A in X is completely characterized by the set of pairs

$$A = (\mu_A(x_i), x_i), x_i \epsilon X \ldots\ldots\ldots\ldots\ldots (5)$$

where $(\mu_A(x_i)$ denotes the grade of membership of x_i in A and is having values in the real interval (0,1).

Let B be a fuzzy set defined in the universe of discourse Y with values in the unit interval, i.e; B: Y → [0,1]. We can use an interesting property derived in previous papers [44] to describe a special assignment:

$$R \parallel A = B \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

where R is a fuzzy relation defined on the Cartesian product X x Y and with values in [0,1], i.e; R: X x Y → [0,1], $\parallel$ indicates inverse fuzzy operator (max, or min). By rewriting equation 6 in terms of fuzzy operators.

$$\bigcup_{x\epsilon X}\left[A(x) \bigwedge R(x,y)\right] = B(y) \ldots\ldots\ldots\ldots (7)$$

for any y ϵ y, where ∪ and ∨ are the classical max and min operators defined by:

$$\mu_{A\bigcup B} = \max\{\mu_A, \mu_B\}$$
$$\mu_{A\bigwedge B} = \min\{\mu_A, \mu_B\}$$

As an example application of the above concept in task assignment, consider the following data:

Given the task - skill requirement matrix

Skills Requirement

Task	Y_1	Y_2	Y_3	Y_4	Y_5
Task-1 (X_1)	0.1	0.2	0	1	0.7
Task-2 (X_2)	0.3	0.5	0	0.2	1
Task-3 (X_3)	0.8	0	1	0.4	0.3

Y_1	=	previous knowledge about the task
Y_2	=	vision requirement
Y_3	=	lifting capability (load or weight)
Y_4	=	judgement requirement
Y_5	=	computational requirement

Skill

Operators	Y_1	Y_2	Y_3	Y_4	Y_5
Operators-1(Z_1)	.9	.2	.8	04	0
Operators-2(Z_2)	0	1	0	.2	1
Operators-3(Z_3)	.3	.8	.7	.3	0
Operators-4(Z_4)	.4	0	1	0	.8

The computational procedure is illustrated below. The matrix R is defined by

$$R(X,Z) = A(X,Y) \,|\, B^T(Y,Z) \ldots\ldots\ldots\ldots (8a)$$

$$R(X,Z) = \max \{\min \left(\mu_A(x,y), \mu_B^T(y,z)\right)\} \ldots (8b)$$

As an example computation of R(x,z); let $X = x_1$, $Z = Z_1$, $Y = \{Y_1, Y_2, \ldots, Y_5\}$

then,

$$\min (\mu_{A(x_1,y_1)}, \mu_{B(y_1,z_1)} = (0.1, 0.9) = 0.1$$

$$\min(\mu_{A(x_1,y_2)}, \mu_{B(y_2,z_1)}) = \min(0.2, 0.2) = 0.2$$

$$\min(\mu_{A(x_1,y_3)}, \mu_{B(y_3,z_1)}) = \min(0, 0.8) = 0$$

$$\min(\mu_{A(x_1,y_4)}, \mu_{B(y_4,z_1)}) = \min(1, 0.4) = 0.4$$

$$\min(\mu_{A(x_1,y_5)}, \mu_{B(y_5,z_1)} = \min(0.7, 0) = 0$$

$$R_{(x_1,z_1)} = \max(0.1, 0.2, 0, 0.4, 0) = 0.4$$

By repeating the procedure for all x and y indexes, the operator - task capability matrix derived is given below:

The assignment problem can now be done with any optimization approach [50]. The Simplest assignment heuristic utilized in TOP is to assign task x to teleoperator z using $\max_Z$ [R(X,Z)] criterion, i.e.;

$$\forall_z \exists_x | x \rightarrow z : {}_z Max[R(x,z)] \ldots\ldots\ldots\ldots\ldots (9)$$

As an example of the above criterion:

for task x_1: max (0.4, 0.7, 0.3, 0.7) = 0.7, thus assign task x_1 to operators z_2 and z_4;

for task x_2; max (0.3, 1, 0.5, 0.8) = 1, thus assign task x_2 to operator z_2;

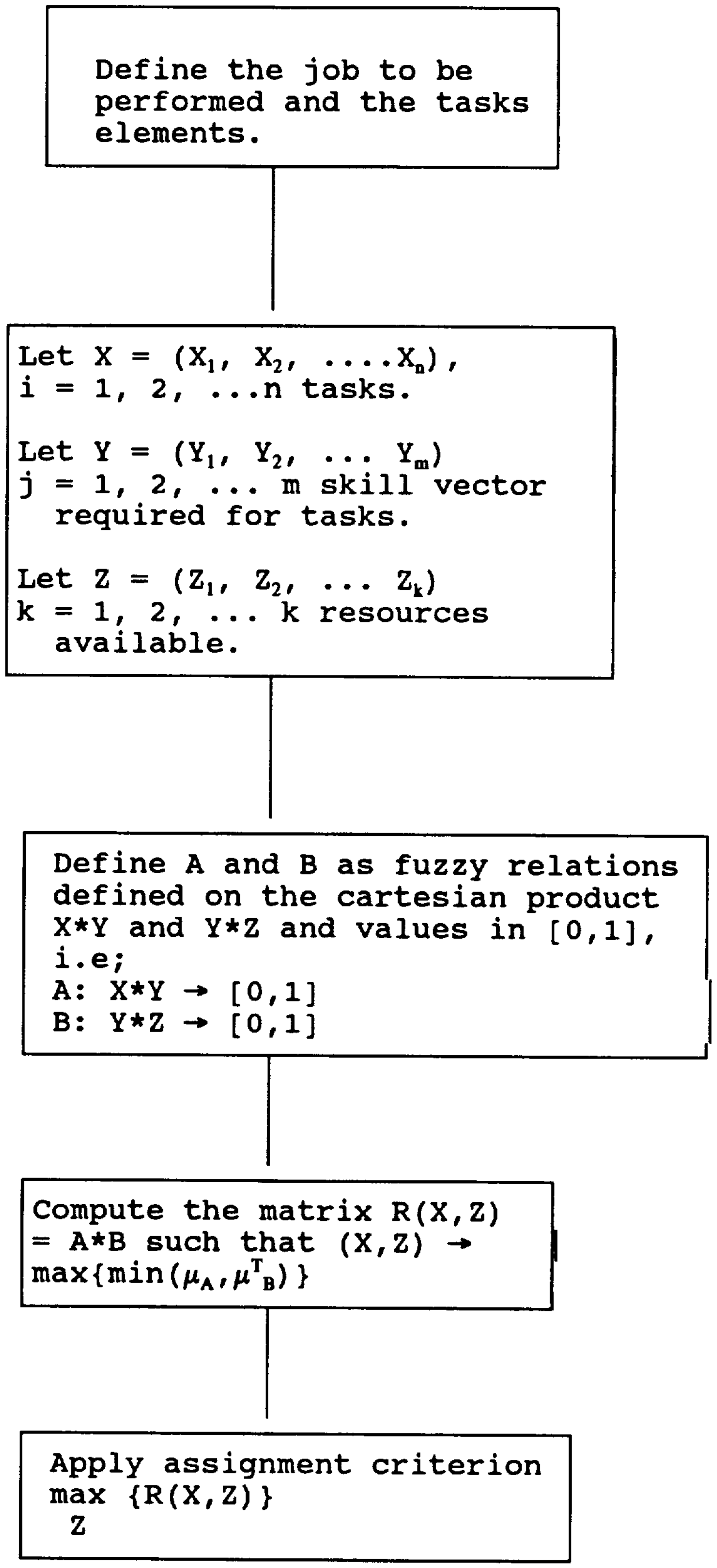
for task x_3; max (0.8, 0.3, 0.7, 1) = 1, thus assign task z_3 to operator z_4.

As the results may indicate, it is possible to use multiple resources for a task. For example, in task X_1, two robots Z_2 and Z_4 performing a conjunctive assembly task may be used based on the fuzzy rankings. Also, the method can be used for technology evaluation for investment purposes. In this trivial example, it is indicative that operator Z_1 may not be useful for any of the tasks. However, if Z_1 is human, then instead of taking part on direct task execution, Z_1 can be viewed as a system supervisor. The computational process of the task-assignment problem is given in Fig. 6.

4.5 SUMMARY

This chapter has discussed the task-planning concepts for a telerobotic environment. The planning formalism is an extension of concepts from previous works on GPS, STRIPS, and ABSTRIPS. However, because of the nature of telerobotics, we have introduced a new planning procedure based on how humans deliberate before making a decision. The Deliberate Plan Network (DPN) presented is an attempt for compact representation of knowledge in this domain. The recursive nature of DPN is such that the concepts of "ADD" and "DELETE" operators from STRIPS are preserved without a particular plan being destroyed during task execution.

The task assignment procedure goes further more than the subjective ratings method proposed by Parker and Pin [50] to consider fuzzy values. Fuzzy task assignments are considered more robust and dynamic in that subjectivity of expert opinions on task difficulty, resource (operator) capabilities and the overall viewpoints on assignment processes are taken into consideration at the planning stage prior to task execution. The demonstration of these concepts will be shown in the later part of this report.



Define the job to be performed and the tasks elements.

Let $X = (X_1, X_2, \ldots .X_n)$, $i = 1, 2, \ldots n$ tasks.

Let $Y = (Y_1, Y_2, \ldots Y_m)$ $j = 1, 2, \ldots m$ skill vector required for tasks.

Let $Z = (Z_1, Z_2, \ldots Z_k)$ $k = 1, 2, \ldots k$ resources available.

Define A and B as fuzzy relations defined on the cartesian product X*Y and Y*Z and values in [0,1], i.e;
$A: X*Y \rightarrow [0,1]$
$B: Y*Z \rightarrow [0,1]$

Compute the matrix $R(X,Z) = A*B$ such that $(X,Z) \rightarrow \max\{\min(\mu_A, \mu^T_B)\}$

Apply assignment criterion $\max\limits_{Z} \{R(X,Z)\}$

Fig. 6: The Computational Procedure For Fuzzy Task Assignment Problem

CHAPTER 5

JOB SCHEDULING IN A TELEROBOTIC SYSTEM

5.1 Teleoperation As A Constraint-Directed Scheduling Problem

In normal scheduling terminology, "jobs" in a telerobotic system are "operations" or "tasks" to be performed. The activities in performing these "operations" are known as teleoperations.

A teleoperation involves multiagents which are supposed to interact symbiotically during task execution. The interdependence of these agents comes about from sharing the same limited resources. Particularly, each of the agents has its own state control variable and goal functions. Therefore, in scheduling a teleoperation, the total amount of the resources available becomes a decision variable itself. Hence, teleoperation is a constraint-directed scheduling problem similar to open job shops as observed by Fox [20].

In addition to the above observations, scheduling teleoperation is also constrained by its unstructured domain. In fact there are at least four scenarios that constitute to this unstructured environment. Some of these are:

(1) Planning teleoperation is context-based; therefore, this type of scheduling decisions made contextually at each level of plan abstraction contextually.

(2) The basis or criteria for optimal scheduling policy are not well defined since the environment is usually unstructured. Here, the central issue is not to optimize a given schedule as but to apply scheduling in manufacturing systems.

(3) There are induced lags or time delays in information flow between the agents. Thus, a plan change during a time lag may change an entire schedule. Of course, rescheduling problems cannot be solved analytically except by a rich domain independent knowledge-based system.

(4) Since scheduling depends on "planned" information, an infeasible plan will usually generate an infeasible schedule. Thus, attention focusing on selecting the "best" schedule for future use in another context may not be practicable.

In recognition these problems, we have developed a heuristic algorithm that deploys planning and scheduling based on task scenario. The idea is to focus attention on imminent (local) plans. When the plans change, the schedule tries to anticipate a new configuration of job sequence based on current resource availability. For each task plan, partial global schedules (PGS) are constructed. A PGS receives plan information from a window with active plan contexts. A window that contains a tentative schedule is matched with the current active plan and a "better" schedule is generated for the situation. A plan window can be viewed as a local hypothesis that tests the availability of resources to a current plan. At the end of each schedule session, the plan is retroactively imbedded along with its schedule context into the global knowledge base for future reference. The situation is depicted pictorially as shown in Fig. 7.

5.2 The Scheduling Algorithm

The scheduling problem can be formulated as follows: given N tasks, each of which requires a certain amount of effort from a labor crew (not necessarily all at once), how should one schedule the tasks to obtain a total work load balance and work cooperation that "minimize" job completion time?

The following definitions are used throughout the discussion:

M number of resources available;

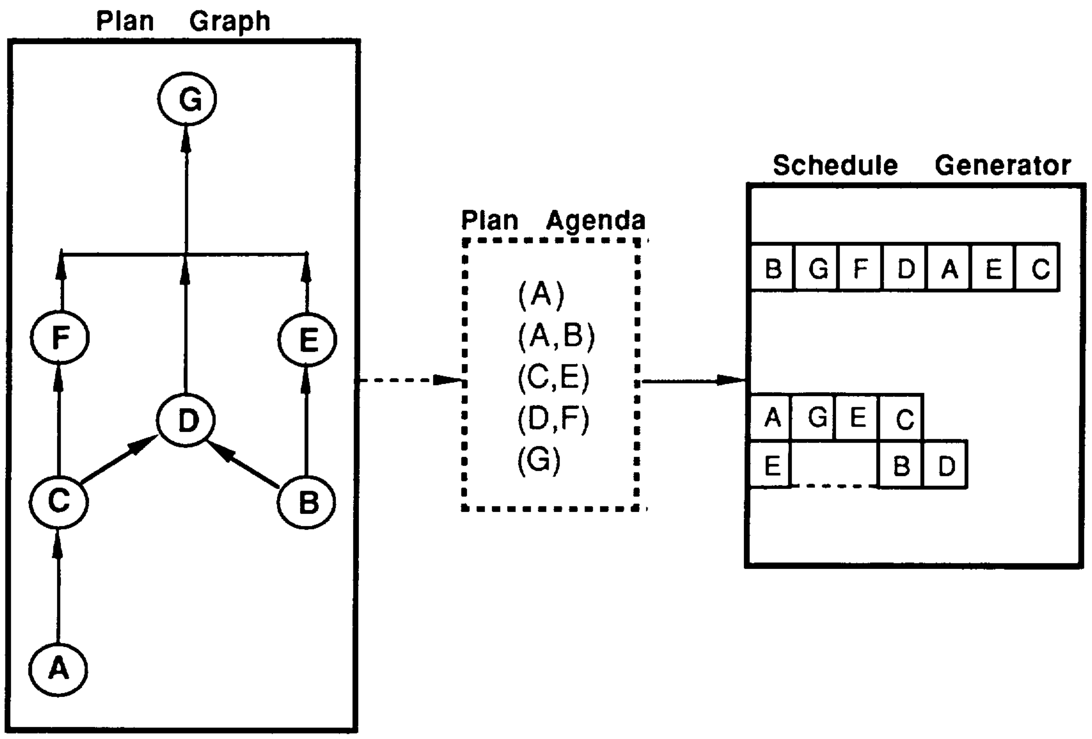


Fig. 7: An Example Partial Schedule Generator From A Plan Graph.

T	a set of tasks;
T_i	a member in T referred to as task i;
R	a set of resources;
R_j	a member in R referred to as resource type j;
$\cup$	a disjunction (OR);
$\cap$	a conjunction (AND);
P	a plan or a sequence of proposed action;
$\exists$	an existential quantifier, "there exist";
$\forall$	a universal quantifier, "for all";
t_i	the processing time of task i;
ϵ	belong to;
k,i,j	indexes or counter notations;
n_1	a set of scheduled tasks;
n_2	a set of unscheduled tasks (or waiting tasks);
<—>	parallel logical notation; for example i <—> j means that i is parallel to j;
N	number of tasks.

Heuristically, the scheduling problem is as follows:

DO: WHILE < PLAN **P** $\neq$ 0 >

1. Select the plan with an immediate goal based on plan priority. Initialize job completion time F(**P**) = 0, and schedule clock, S=0.

2. **IF** the cardinality of **R** $\geq$ cardinality of **T** (i.e. if $M \geq N$) then schedule all tasks simultaneously. Let $n_1 = n_2 = 0$; GO TO step 10.

3. **Else**
Select a task with available resources and store in n_1 and tasks without resources into n_2 (See e.g.; Fig. 8). Set the completion time of all jobs in n_2 to a very large value to indicate their low priorities. That is, $t_j = \infty$, for all $j \in n_2$.

4. Schedule all tasks in n_1 and update their completion times.

5. Select tasks in n_2 for processing based on the most available resources i.e;

$$\exists_k, \exists_x (k \epsilon n_2, x \epsilon n_1 ; k \epsilon [\forall_x \min \{t_x\}]) \ldots\ldots\ldots\ldots (10)$$

Equation (10) states that the tasks in n_1 with the minimum processing time will release resources for unscheduled tasks k currently in n_2.

6. Update the schedule clock to the current end of event (first task completion) and start of a new schedule from n_2. That is

$$S = S + Min \ \{t_x\}, (see\ step\ 5) \ldots\ldots\ldots\ldots\ldots\ldots (11)$$

7. Remove x from n_1 since this task has been completed; add k to n_1 to update a list of task execution in progress; and update n_2 by deleting k which is now scheduled. That is;

$$n_1 = n_1 + k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (12)$$

$$n_2 = n_2 - k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (13)$$

8. **If** all jobs for the current plan **P** have been scheduled (i.e.; n_1 = N and n_2 = o) then go to step 10.

9. **Else** go to step 5.

10. Calculate the job completion time for plan P. This is given by:

$$F(P) = MAX \begin{cases} \forall_{i \epsilon N} \{\max(t_i)\},\ if\ n_2 = 0,\ see\ step\ 2 \\ \forall_{i \epsilon n_1} \min(t_i) + Max\ \{\exists_{j \epsilon n_2} [j \epsilon i \bigcup_{j \to i} | j \to \{\min(t_j)\}] \end{cases} \cdots (14)$$

Equation 14 states that if resources are available at the beginning for all tasks as defined in step 2, then F(P) is the time to complete the longest task. Otherwise, F(P) is the addition of the start times of all active jobs (defined by minimum time of all tasks already completed in n_1,) and the expected maximum processing time of all jobs in n_2 which can either be processed in parallel (i<->j) or j is immediate follower of i with the minimum processing time. Figures 8-10 are use to illustrate the above concepts.

Note that R_2 and R_3 can be robots while R_1 is the human. As shown in Fig. 8B, tasks T_4 and T_5 are in list n_2 waiting to be scheduled. In this example, there are at least two ways to deploy the available resources (see Fig. 8C). The decision to use R_2 along with R_1 in task T_4 (Fig. 9A) or use R_2 along with R_3 in executing task T_5 (Fig 9B) depends on the executing of the fuzzy task assignment model discussed in Chapter 4 of this report. In all cases, if we assume no inserted time delay, a minimum schedule turnaround time (see Fig. 10) can be achieved. It should be recognized that in the real situation, minimization of teleoperator time delay is a factor that should not be discounted. This is taken into consideration during the task execution model.

5.3 SUMMARY

We have presented a heuristic technique for scheduling limited resource teleoperations. We view a teleoperation as a constraint-directed scheduling problem in an unstructured environment. The interdependence of teleoperators and their various state controls are considered as the major decision variables in the scheduling policy. Although no sets of optimization criteria have been defined for the scheduling problem, the algorithm is general enough to incorporate time lags between teleoperators during task executions. Since the scheduling formalism incorporates information directly from the planner, it is easy to preempt

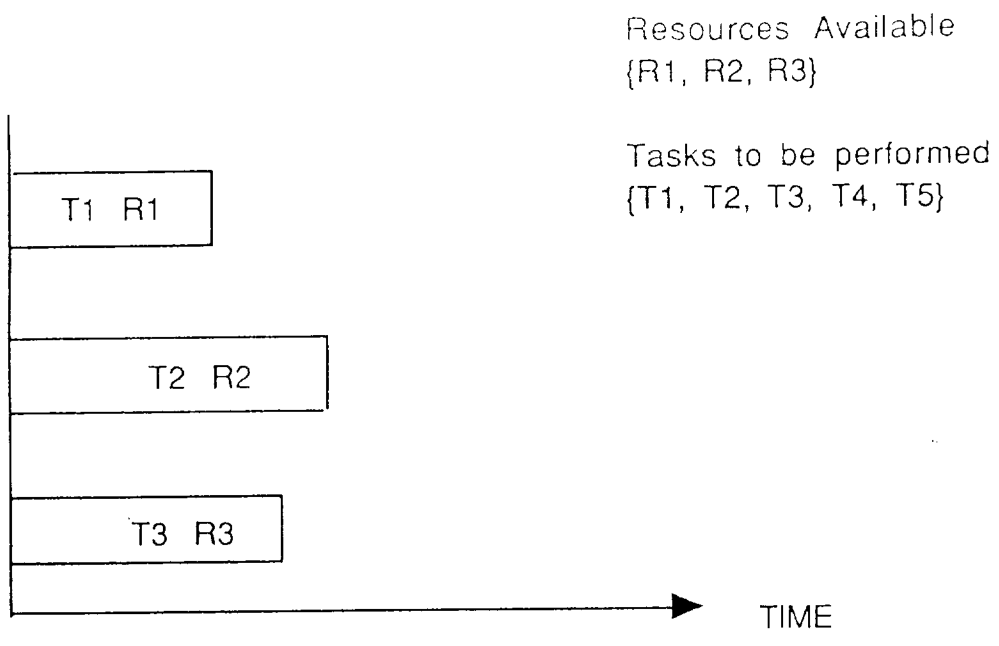


FIG. 8A Beginning Schedule

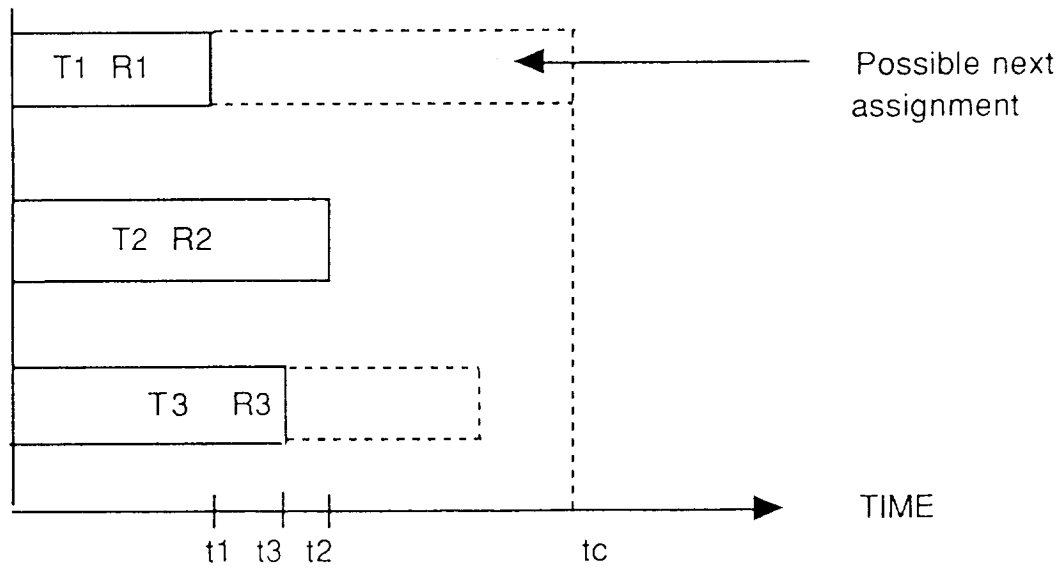


FIG. 8B A Schedule with tasks T1 and T3 completed at times t1 < t3. Tasks T4 and T5 are to be scheduled next. t_c = potential Job Completion time.

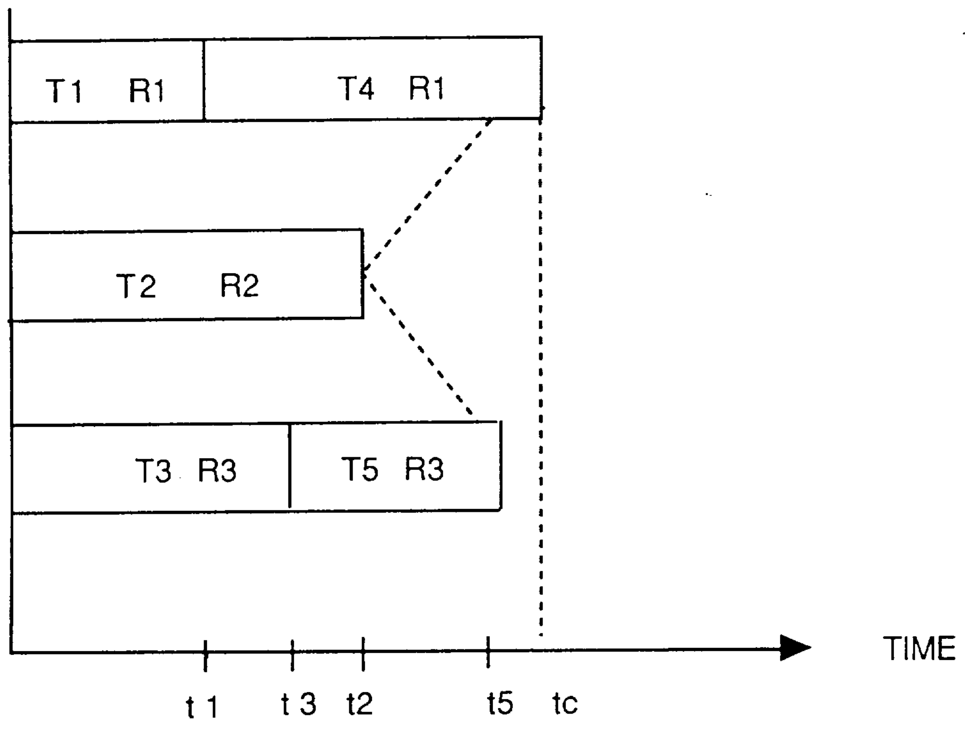


FIG. 8C A Schedule showing a dynamic availability of resource # R2.

R2 can be deployed in any of the two scenarios in Fig. 9A and 9B.

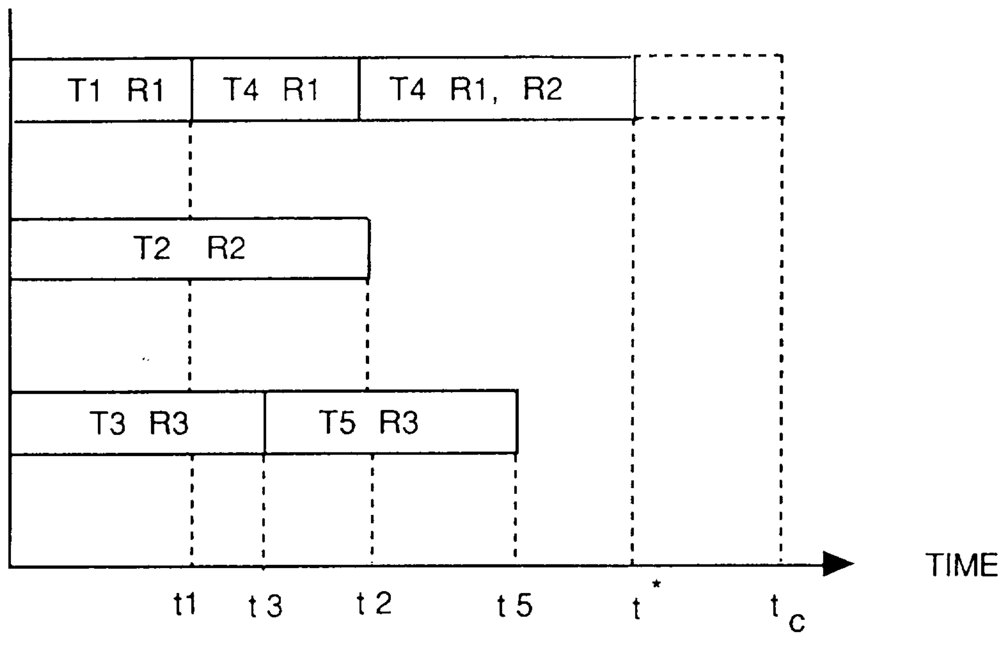


FIG. 9A The deployment of R2 in task T4 reduce, t_c to t^* (assume no delay time).

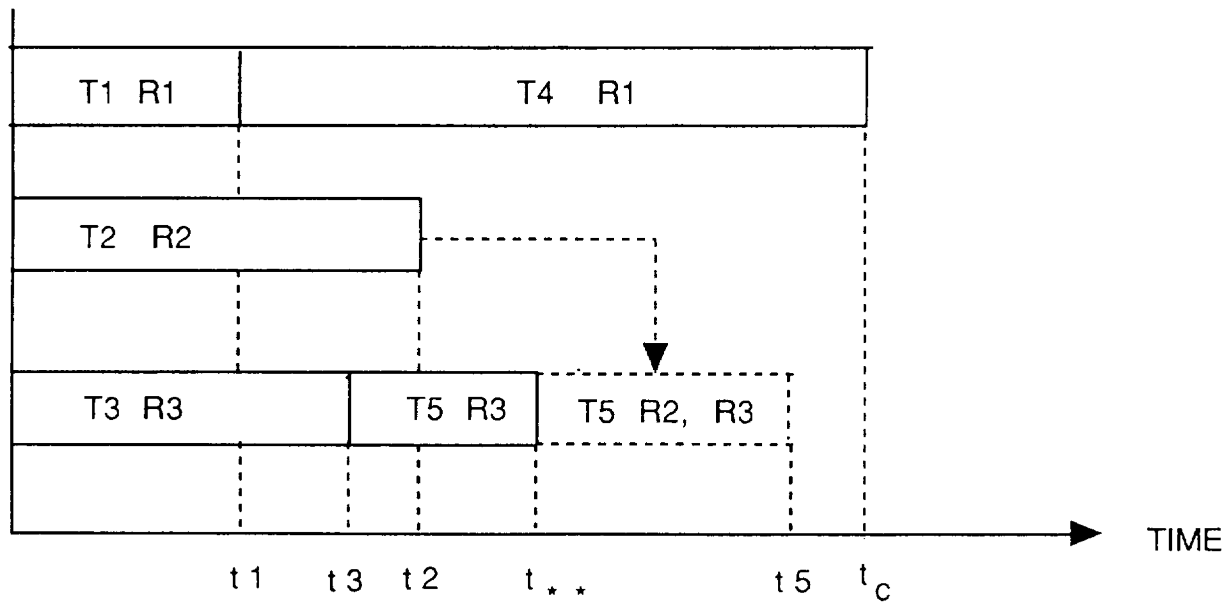


FIG. 9B The deployment of R2 in Task T5 reduces t5 to t_{**}.

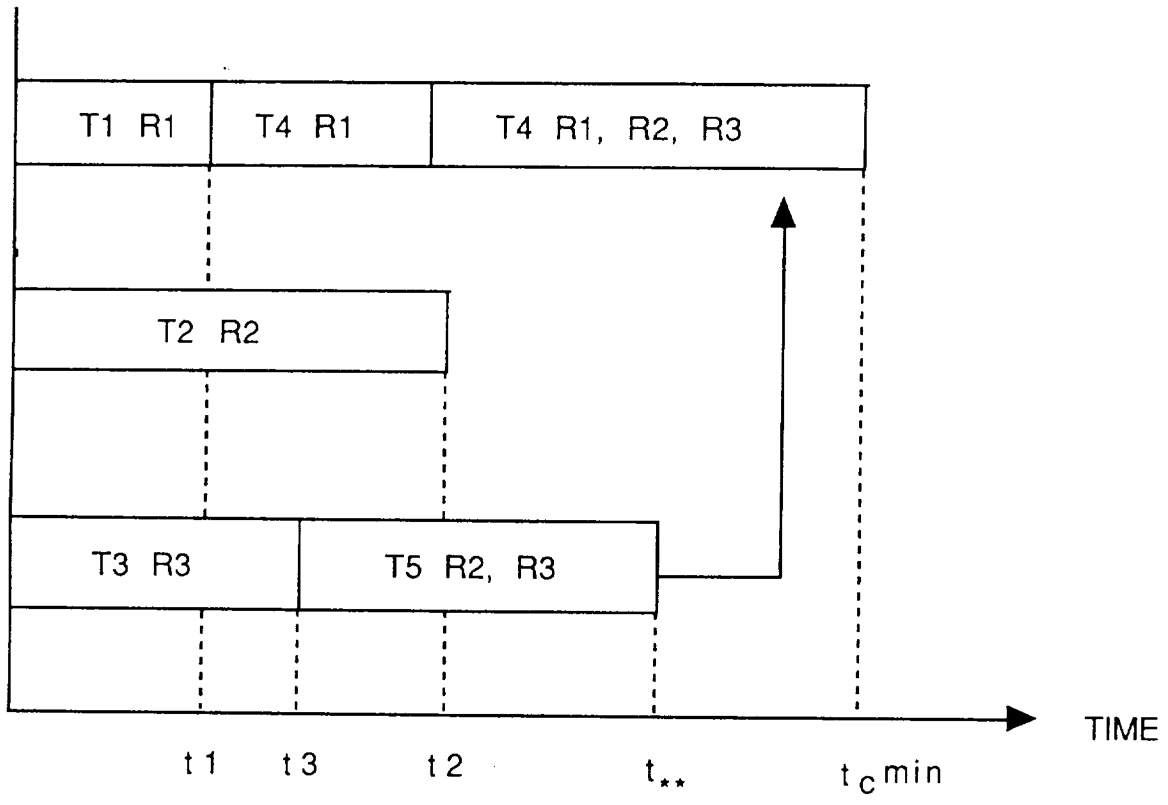


FIG. 10 A possible minimum time schedule with no inserted idle time t_c min < t_c .

current schedules in order to respond to plan changes. This feature of the algorithm makes it to be domain-independent suitable to any planning involving multiagent systems.

CHAPTER 6

MODEL APPLICATIONS AND VERIFICATIONS

6.1 The Application Environments

The TOP, which is the main architecture for planning and scheduling have been successfully applied to two scenarios. In the first scenario, TOP is used in an aircraft turnaround domain. The applications are achieved by using a NEXPERT™[3] expert system shell. In the second scenario, we apply TOP to a simulated block world where constraints are adaptively posted [43]. This example is implemented in a MULISP™ domain.

6.2 Aircraft Turnaround Function

Aircraft turnaround function consists of refuelling, re-arming and minor aircraft maintenance during combat missions. When an aircraft is returning after delivering a sortie, the pilot relays the status of the aircraft to the ground crew in order to give them a head start on preparations for turnaround. The crew chief gives thorough visual examination of the aircraft based on the pilot's information. If the aircraft has no damage, it is taxied to the turnaround area where the turnaround functions are executed.

During combat, potential exposure of crew members to hostile situations cannot be avoided. The use of telerobotic assistance will result in a smaller crew size for the aircraft turnaround function. The critical goal is to reduce the total man hours

[3] NEXPERT™ is a treadmark of Neuron Data, Inc.

expended in the turnaround operations. We use the telerobotic planning and scheduling model to test the feasibility of teleoperation in an aircraft turnaround domain.

6.2.1 Planning Aircraft Turnaround Function

The turnaround function chosen for demonstration is aircraft refueling. Refueling is the most commonly performed turnaround function and requires the least amount of dexterity, which makes it the most amenable to robotics.

At the planning phase, the turnaround function tasks are represented in the form of a data base (See Table-1). The planner uses the information on a job code, for example AG2, to generate a plan tree as shown in Fig. 11. As indicated in Fig. 11, the bold lines indicate a possible feasible plan path. This is achieved by heuristic reduction using the plan deliberation graph (PDG) discussed earlier. Note that Table-1 is obtained by a task analysis of all jobs to be performed on each of the nine aircraft stations (See Exhibit 4).

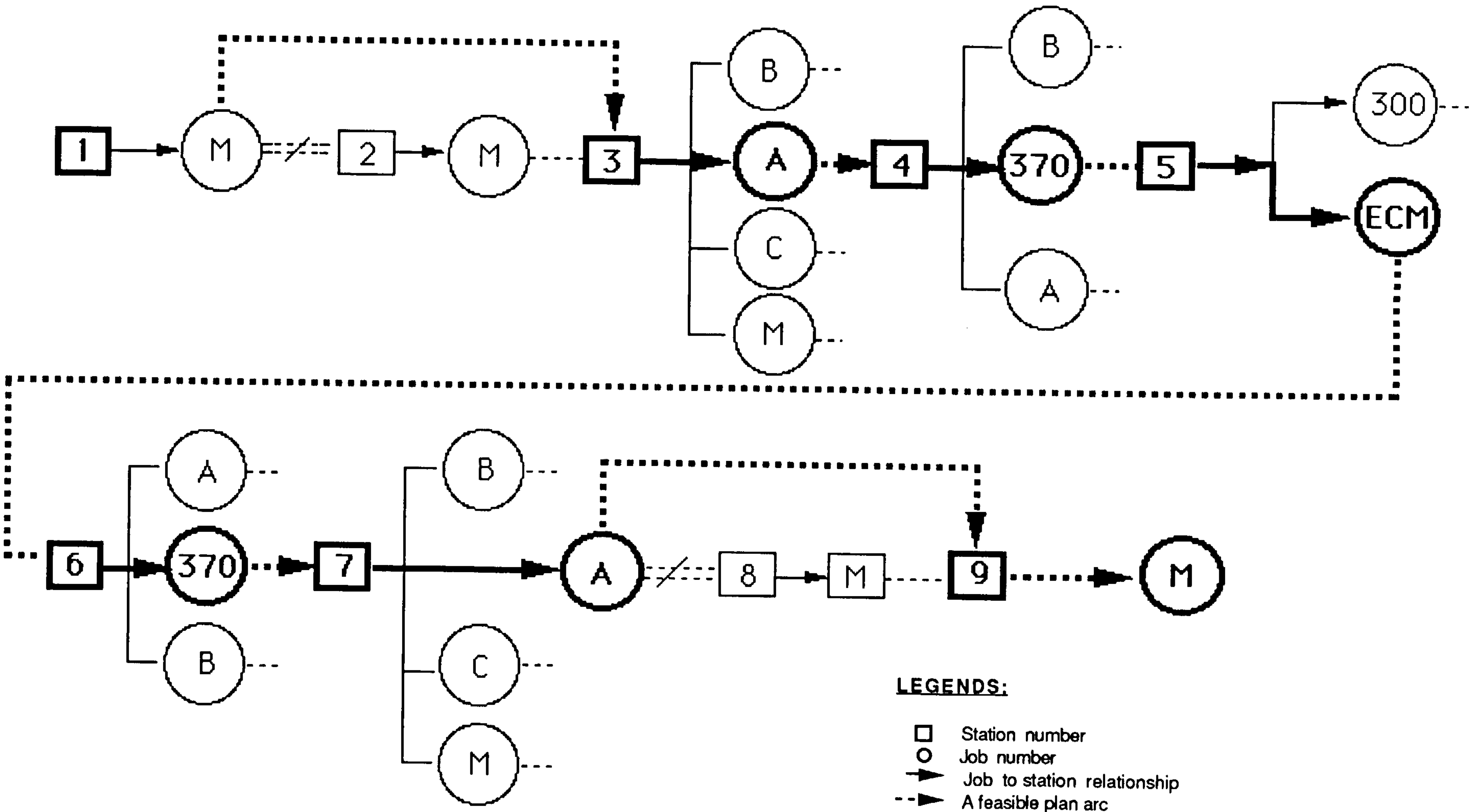
The task planner TOP is written in DBASEIII™ with Artful™ interface. By entering a configuration code, the planner generates a list of tasks (with the task name, station to be worked on) and the expected time required to complete a task by either a human or a robot. In TOP, a configuration code defines a macro job. A final plan graph for job AG2 is shown in Fig. 12. As the final plan graph shows, the task structure indicating precedent constraints for a job A(3) is indicated. A(3) means that an MK82 missile is to be loaded in station 3. Similarly, for aircraft refueling, 370(6) means that a fuel tank mounted on station 6 is to be refueled.

Table-1: A list of possible configurations

Config Code*	9	8	7	6	5	4	3	2	1
AA1	M	M	M	-	EMC	-	M	M	M
AA2	M	M	M	370	300	370	M	M	M
AA3	M	M	M	-	-	-	M	M	M
AG1	M	-	A	A	300	A	A	-	M
AG2	M	-	A	370	ECM	370	A	-	M
AG3	-	-	B	B	ECM	B	B	-	-
AG4	M	-	C	-	300	A	A	-	M
AG5	M	-	C	370	300	370	C	-	M
AG6	M	M	B	B	300	A	A	M	M

A - 3 MK82'S MOUNTED ON A TER
B - MK84
C - 2 AGM-65'S MOUNTED ON A LAU-88 LAUNCHER
M - AIM-9L MISSILE MOUNTED ON MISSILE LAUNCHER
370 - FUEL TANK MOUNTED ON STATIONS 4 & 6
300 - FUEL TANK MOUNTED ON STATION 5

Fig. 11: An Example of Plan Generation.



LEGENDS:

□ Station number
○ Job number
⟶ Job to station relationship
- -▶ A feasible plan arc
=/= Infeasible plan arc
- - - Continue to explore plan arcs

The feasible plan for configuration code AG2 is:
M(1), A(3), 370(4), ECM(5), 370(6), A(7), M(9)

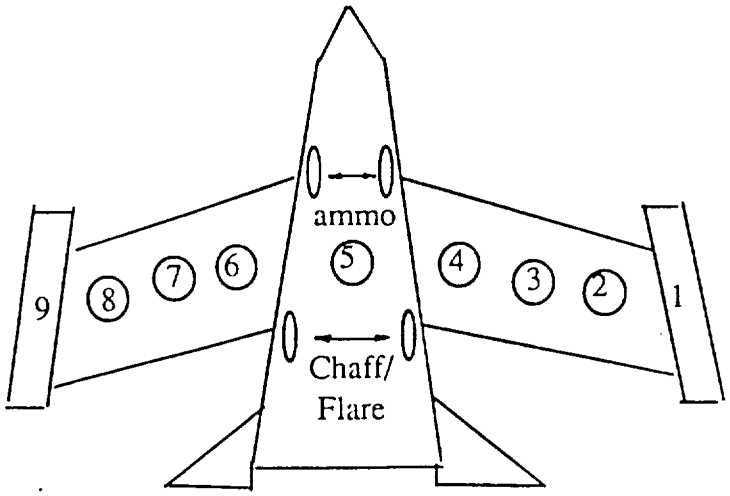
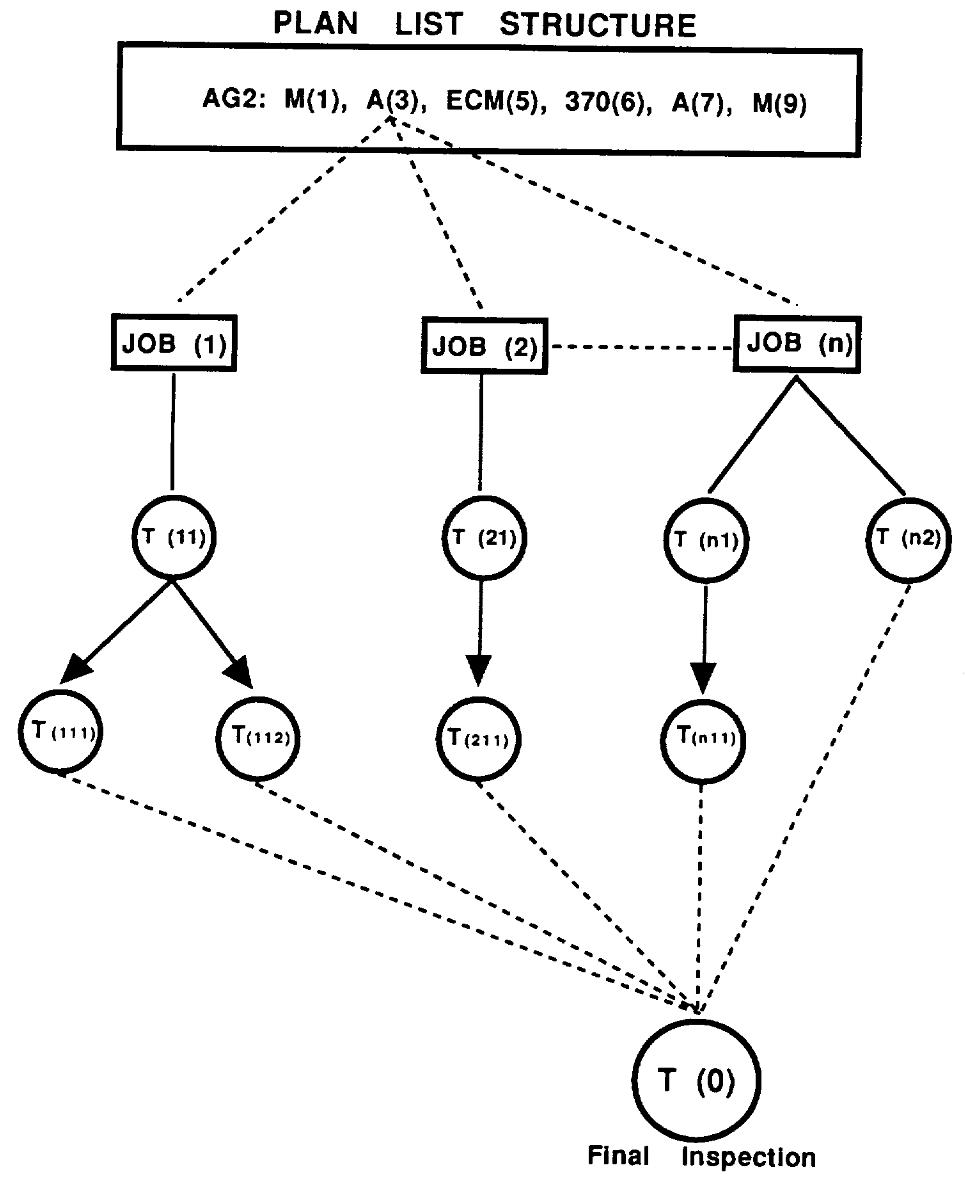


Exhibit 4: Aircraft Work Stations



PLAN LIST DEFINITION

AG2: Is configuration code
= ICT plan.

J(S): Job to be performed
on station S.
e.g. A(3) is load 3 MK82's
on station 3.

JOB DEFINITION

Job (i) : Job i
T(i..) : Task or subtask for job i.
T(0) : Final check.
E.g. For A(3) plan
J(1) = MER prepration
J(2) =Loading, etc.

Fig. 12 : A TREE-LIKE TASK SYSTEM FROM TOP

6.2.2 Scheduling Aircraft Turnaround Function

Upon the completion of planning, the TOP scheduler picks the job with highest priority for scheduling. The scheduler consults the task skill matrix to verify each task difficulty and their respective fuzzy ranking? Using the fuzzy attributes and structural constraints, the individual task is assigned to either a robot or human for execution. In cases where the robot executes a task, the human operator serves as a control supervisor while the task is autonomously executed under a sensor-driven control procedure.

The scheduling procedure is implemented with NEXPERT™. In NEXPERT, two main characteristics of objects can be distinguished: what they represent and how they should be used by the system. The structure of an object is as follows:

name
classes
subobjects
properties

As an example, consider a turnaround function plan P1 with all the jobs to be performed: This can be represented as

```
(    CNAME = P1
   (@CLASS = job
       (@ PROPERTIES =
     endtime         "job end time"
     id              "job ID"
     job_name        "job name"
     personnel1      "crew name:
     personnel2      "crew name"
     personnel3      "slot for floating Crew"
     starttime       "job start time"
     station         "station location"
     time1           "time for human"
     time2           "time for robot"
     timepd          "dynamic slot"
     unassigned      "logical variable to assign task"
     working-on )))
```

Thus, each job category has start and end times, dynamic slots to hold a vector of unassigned jobs, crew members, a job identification and the station number to be worked on. Generally, NEXPERT knowledge representation allows for dynamic objects. Dynamic objects act as instances of variables which are those classes and objects. Examples of dynamic objects are "personnel13" and "timepd" slots for holding non-instantiated variables.

A very important aspect of the object organization is the notion of inheritance, that is to say, the way objects and classes can communicate values to each other. An object will typically be able to inherit a value of one of its properties from one of its classes or parent objects. For example, the door of the car will inherit the "color" of the car (parent object). All these value passing mechanisms, which represent types of default reasoning, are customizable at the lower level of granularity: the property of the object or the class. Objects also inherit functions, or methods.

During a job schedule, the knowledge processing environment dynamically creates a "Node" for each personnel resource and the class attributes as discussed earlier. Dynamic node creation allows for possible job pre-emption, resumption and assignment of idle resources.

At the end of each schedule, the system records the generated feasible schedule into a separate external dBASE file. This data remains available for later analysis.

The scheduling process uses the schedule algorithm of Chapter 5 (Section 5.2). The invocation of the algorithm uses rules to execute each schedule profile. For example, to select a plan for a schedule, Rule 85 below is fired first by setting a priority (INFACT) to the highest value possible.

```
(@RULE=   R85
     @INFCAT=100;
     (@LHS=
           (Yes (GetData)))
     (@HYPO=    (GoGet)
     (@RHS=
          (Retrieve ("c:\dbase\joblist.dbf")
(@TYPE=DBF3;@FILL=ADD;@NAME="' job_ '!id!";\
@CREATE= |job|, |timecode| '@PROPS=id, job_name,\
station, unassigned,time1,time2,starttime,\
endtime;@FIELDS="id","job_name","station",\
"unassigned","time1","time2","starttime",\
"endtime";))
           (Do    (MAX(<|job|>.id))       (n))
           (Do    (n)    (ctr)
           (Do    (1)    (wpcnt))
           (Do    (1)    (chcnt))
           (Retrieve    ("c:\dbase)upldstat.dbf")
(@TYPE=DBF3; @FILL=ADD;@NAME="' station'!id!";\
@CREATE=|station|;@PROPS=free;@FIELDS="station";\))

           (Let   (wp1.free)    (TRUE))
           (Let   (wp2.free)    (TRUE))
           (Let   (wp3.free)    (TRUE))
           (Let   (<|job|>.working_one)           (FALSE))
           (Do    (0)   (timeclock))
           (Let   (ch1.free)    (TRUE))
           (Let   (ch2.free)    (TRUE)))
```

A list of unassigned tasks can be constructed using the global rule:

```
          (@RULE=   R83
                (@LHS=
                     (=    (ctr)       (0))
                     (>    (SUM(<|job|>.unassigned))       (0))
                     (Is   (<|job|>.working_on)        (FALSE)
          )
          (@HYPO=    find_smallest_timepd1)
          (@RHS=
                (Do     (99999)    (<|job|>.timepd))))
```

The task assignments are performed asynchrously using the global rule:

```
(@RULE=   R74
     (@LHS=
          (>     (ctr)          (0))
     )
     (@HYPO=    continue_assign)
     (@RHS=
          (Reset    (continue_assign))
          (Do  ('job_'\ctr\.job_name)     (current_job.job_name))
          (Do  ('job_'\ctr\.station)      (current_job.station))
          (Do  ('job_'\ctr\.unassigned)  (current_job.unassigned))
          (Do  ('job_'\ctr\.id)      (id))
          (Do  ('ctr-1)     (ctr))))
```

Finally, the global rule to check task completion and update system status is given by:

```
(@RULE=   R79
     @INFCAT=100;
     (@LHS=
          (=    (ctr)       (0))
          (=    (SUM(<|job|>.unassigned))       (0))
     )
     (@HYPO=    done)
     (@RHS=
          (Do   (MAX(<|job|>.endtime))      (timeclock))
          (Write     ("c:\dbase\finjob.dbf")
(@TYPE=DBF3;@FILL=NEW;@PROPS=starttime,endtime,\
job_name,name,station;@FIELDS="STARTTIME",\
"ENDTIME","JOB_NAME","NAME","STATION";@ATOMS=<|personnel|>;\
))
         (Write    ("c:\dbase\jobdone.dbf")
(@TYPE=DBF3;@FILL=NEW;@PROPS=job_name,starttime,\
endtime,id,station,time1,time2,unassigned,\
personnel1,personnel2,personnel3;@FIELDS=JOB_NAME",\
"STARTTIME","ENDTIME","ID","STATION","TIME1",\
"TIME2","UNASSIGNED","PERSONNEl1","PERSONNEl2",\
"PERSONNEl2";@ATOMS=<|job|;))))
```

Table 2 contains a sample schedule generation for eleven tasks and three personnel. Table 3 shows sample task assignments for each of the three personnel. Note that the job schedules take place simultaneously with job execution. The execution of jobs by the robot is discussed below.

TABLE 2: Scheduling List for Each Job (Time in Minutes)

Job Names	Station	Start	End	Personnel		
LOAD AIM-9L MISSILE	1	5.00	10.00	wp1	wp2	wp3
UNLOAD TER RACK	3	19.00	21.00	wp2	wp3	
UNLOAD LAU88 & AGM-65	3	21.00	26.00	wp2	wp3	
UNLOAD TER RACK	4	17.00	19.00	wp2	wp3	
LOAD 370 & REFUEL	4	19.00	23.00	ch1	ch2	
REFUEL 300	5	12.00	13.00	ch1	ch2	
LOAD 370 & REFUEL	6	8.00	12.00	ch1	ch2	
LOAD 3 AGM-65	7	10.00	17.00	wp1	wp3	
LOAD AIM-9L MISSILE	9	0.00	5.00	wp1	wp2	wp3
SAFETY CHECK	0	3.00	8.00	ch1	ch2	
LOAD CHAFF/FLARE	0	0.00	3.00	ch1	ch2	

TABLE 3: Scheduling List for Each Personnel (Time in Minutes)

Job Names	Station	Start	End	Personnel
LOAD AIM-9L MISSILE	9	0.00	5.00	wp1
LOAD AIM-9L MISSILE	1	5.00	10.00	wp1

Scheduling List for Each Personnel

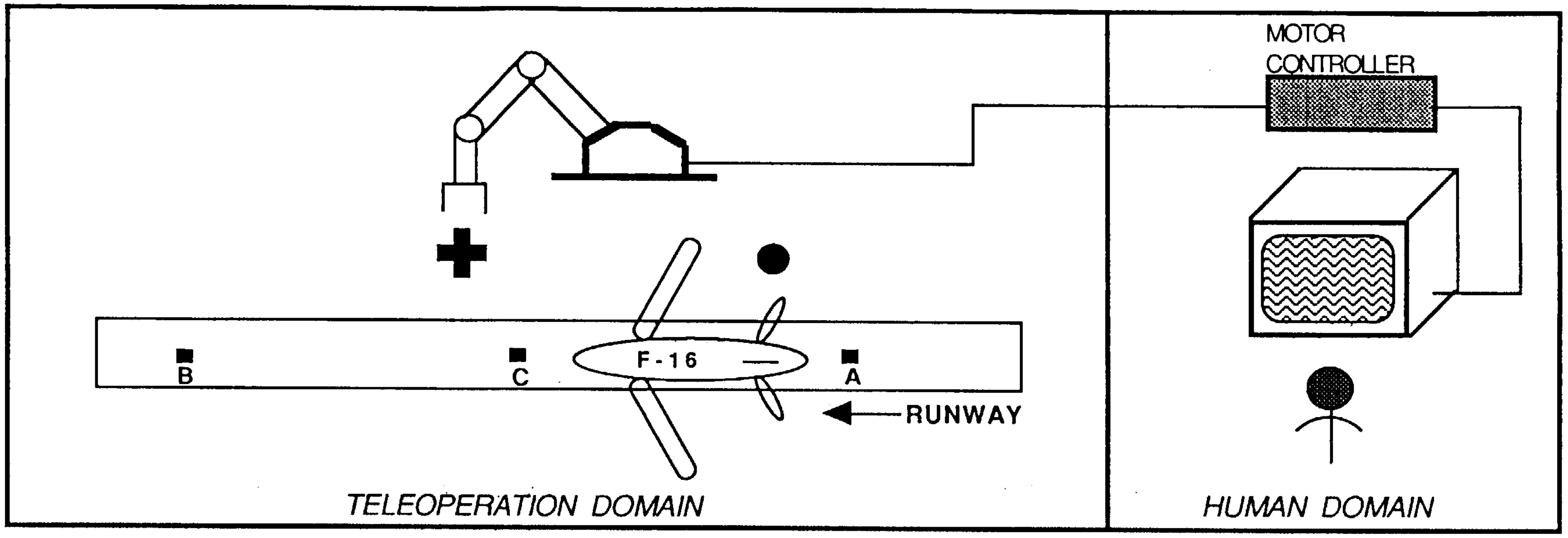
Job Names	Station	Start	End	Personnel
LOAD AIM-9L MISSILE	9	0.00	5.00	wp2
LOAD AIM-9L MISSILE	1	5.00	10.00	wp2
LOAD 3 AGM-65	7	10.00	17.00	wp2
UNLOAD TER RACK	4	17.00	19.00	wp2
UNLOAD TER RACK	3	19.00	21.00	wp2
LOAD LAU88 & 3 AGM-65	3	21.00	26.00	wp2

Scheduling List for Each Personnel

Job Names	Station	Start	End	Personnel
LOAD AIM-9L MISSILE	9	0.00	5.00	wp3
LOAD AIM-9L MISSILE	1	5.00	10.00	wp3
LOAD 3 AGM-65	7	10.00	17.00	wp3
UNLOAD TER RACK	4	17.00	19.00	wp3
UNLOAD TER RACK	3	19.00	21.00	wp3
LOAD LAU88 & 3 AGM-65	3	21.00	26.00	wp3

6.2.3 Robot Execution of Turnaround Function

The task execution simulator is comprised of a five axis robot mounted on a linear slide. This enables the robot to travel the full length of the runway where the turnaround function can be accomplished (See Fig. 13).



A, B, C = Sensors connected to Motor Controller.

✚ = Turnaround area for bombs, missiles and ammo.

● = Refueling area.

Figure 13 : TOP Simulator Cell.

The runway is equipped with a feedback sensor so that the exact location of the aircraft can be given to TOP enabling the robot to perform the functions. There are three equally spaced dip switches on the runway, which when depressed by the wheels of the aircraft, activate them and the position of the aircraft can be given to the minicomputer. The system is interfaced to the minicomputer by a RS232 serial cable through a motor mover, which controls all the stepper motors required to run the model. The description and purpose of the dip switches are as follows:

"A": Refueling area of the aircraft.

"B": Loading and unloading area for bombs, missiles and ammunition.

"C": When activated, gives a clear signal for the next aircraft to be serviced.

The commands used to move the robot are simple natural language commands. These commands control the communication between the robot and the system. As an illustration, the following program syntax is used to move the robot from home position to point "A" on the linear slide. The robot moves its shoulder and wrist to grab the refuelling hose and then goes back to its home position at the end of the runway:

```
(DEFUN  SEND-COMMAND  (A B)
; Formats string commands sent to the Microbot devices
   (COM-OUTPUT-STRING A B)
   (COM-OUTPUT-BYTE 13 B)
   (HANDSHAKE B))

(DEFUN HANDSHAKE (N)
; Receives handshake signals (0,1,or 2) from the Microbot devices
   (SETQ TEST  (COM-INPUT-BYTE (N))
   (COND
     ( (EQUAL TEST 49))
     ( (EQUAL TEST 48)
       (PRINC "INVALID COMMAND") (TERPRI))
     ( (EQUAL TEST 50)
      (PRINC  "STOP  or MODE  ENTERED .   .   .Check     STOP
button") (TERPRI))
    (T (HANDSHAKE N) ) ) )

(DEFUN START  ()
; Initialize the Microbot Teachmover
   (CLEAR-SCREEN)
   (MAKE-WINDOW 4 24 14 44)
   (FOREGROUND-COLOR 15)
   (BACKGROUND-COLOR 1)
   (BORDER-COLOR 9)
   (PRINC "TYPE ' FUEL' TO REFUEL  ——> ")
   (SETQ RES (READ))
   (TERPRI)
   (PRINC " ** Press RESET/RUN button on the Motor Mover") (TERPRI
2)
   (PRINC  "  **  Put  Teachmover  Teach  pendant  in  TRAIN  mode  ")
(TERPRI 3)
   (PRINC " ** PRESS 'C' TO CONTINUE ")
   (SETQ RES (READ))
   (COM-ININ 0 9600 N 8 1)
   (SEND-COMMAND "@DELAY 70" 0)
   (SEND-COMMAND "@ARM #" 0)
   (CLEAR-SCREEN) (REFUEL)
)

(DEFUN REFUEL ()
;Defines the refueling function of the aircraft at POINT 4, 5, 6.
 (SETQ *PRINT-ESCAPE* T)
    (CALIBRATE-HAND) (AIRCRAFT-SERVICE-PLACE)
    (FUEL_HOSE_LOCATION) (FL) (P5)
    (MOVE_FOUR-INCH) (SEND-COMMAND "@STEP 150,0,0,50" 0) (PAUSE-1)
    (MOVE_BACK) (BSFL) (ROBOT_GO_HOME_ (PAUSE-2)
    (AIRCRAFT-READY) (SYSTEM)
)

(DEFUN FL ()
;THIS IS THE FUEL HOSE LOCATION ON THE TARMAC.
   (SEND-COMMAND "@DELAY 25" 0)
   (SEND-COMMAND "STEP 225,0,0,0,0,0,900" 0)
   (SEND-COMMAND "@CLOSE" 0) (SEND-COMMAND "@RESET" 0)
   (SEND-COMMAND "@STEP 225,0,0,-596" 0)
   (SEND-COMMAND "@STEP 225,0,0,0,0,0,900" 0)
   (SEND-COMMAND "@CLOSE" 0) (SEND-COMMAND "@RESET" 0)
   (SEND-COMMAND "@STEP 150,0,0,0,0,0,800" 0)
   (SEND-COMMAND "@STEP 225,0,0,0-276,-276" 0)
 (SEND-COMMAND "@CLOSE" 0))
```

```
(DEFUN FUEL_HOSE_LOCATION ()
'THIS MOVES THE ROBOT TO THE LOCATION OF THE FUEL HOSE ON THE
TARMAC.
(SEND-COMMAND "#5STEP 225,0,0,0,-1686" 0))

(DEFUN ROBOT_GO_HOME ()
;ACTIVATES THE ROBOT TO GO TO ITS HOME POSITION.
(SEND-COMMAND "#GOHOME 225,,,,1" 0))
```

An execution command is defined as a macro syntax. For example to load a bomb from BL1 location on station 2 of the aircraft and wait for further instructions, the following command is issued:

COMMAND > LOAD,BOMB,BL1,STATION2,GOHOME,WAIT.

In general the execution commands are macro operators such as:

LOAD (Bomb and missile), **UNLOAD** (Bomb and missile), **GRASP, RELEASE, AIM7, GOHOME, BASE-LEFT, BASE-RIGHT, SHOULDER-UP, SHOULDER-DOWN, ELBOW-UP, ELBOW-DOWN, PITCH-UP, PITCH-DOWN, ROLL-LEFT, ROLL-RIGHT, WAIT, REFUEL.**

6.3 A Microblock World Teleoperation

The next application of TOP is the constrained microblock world problem. The ordinary microblock problem has been solved as a "pick" and "place" reconfiguration design plan by such systems as STRIPS [16] and ABSTRIPS [17].

In a telerobotic system, the microblock constraints are more than the usually assumed structural relationship (i.e.; the order in which the blocks should be arranged). The problem is over constrained by simulating the environment of the operation: painting blocks to different colors, controlling the intensity of light in the environment, and varying the block configuration with respect to geometric sizes.

The simulated environment consists of one human as a supervisor and a single one-arm microbot robot. Based on the constraint environment, a task-skill matrix was generated and the individual tasks assigned either to the robot or the human.

The simulation begins with an initial configuration of three blocks and allows the user to restack at least two of the blocks and no more than three. Various initial block configurations are randomly generated or defined by the user. Assignment of a task to a resource depends on the task-skill matrix selection criteria. It could be minimum completion time or a minimum expected risk criterion.

Once the task plan is generated, the plan is executed. This is done by passing the subtask sequence to the program operators. Next, an evaluation of the task plan takes place which is in the form of constraint management. The program, upon successful task design and execution, displays the next task to be performed by backtracking through the task-frame hierarchy. The simulation is repeated by changing task execution parameters using internal "seeding" (i.e., random numbers are generated by different random seeds).

6.3.1 Microblock Simulation Results

Twenty simulation runs were replicated for each task to be performed. Each sequence of tasks used a different same random seed for randomized design of the job assignment profile. The results of the experiments are shown on Tables 4-11. The results show the task completion times for both human and robot teleoperators respectively.

In Table 4, for example, during the first run, the human operator took 5.88 seconds to complete the task. This is a special example in which the task skill matrix was rated such that the task took place in a "hazardous" environment. The average completion and standard deviation times are also given on Table 4. Figures 14-17 show the plots of these task times for each teleoperator. It is difficult to say precisely which of the operators is better since the environments in which tasks are performed change with respect to simulated conditions. Also, the frequency with which the teleoperators are assigned to a task see Tables 7-10 depend on the teleoperator rating from task-skill matrix model as well as the simulated environment. Figures 18-21 show corresponding frequency plots. For all practical definitions of a teleoperation, it is better to minimize task assignments to the human especially in high risk environments. Another observation from the task assignment frequency plot is that these values can be used to predict the assignment of tasks. For example, to remove block-2 from block-1 (in the environment which the task is performed), we can predict that 20% of the human effort and 80% of the robot effort will be required.

Table 4 TASK: REMOVE BLOCK-2 FROM BLOCK-1

NO. OF RUNS	HUMAN TIME	ROBOT TIME
1	5.88	0.907
2	0.872	0.864
3	0.144	1.06
4	3.36	0.712
5	3.88	1.1
6	3.72	1.22
7	3.00	0.943
8	2.93	0.101
9	0.515	1.32
10	0.936	0.99
11	1.41	1.18
12	2.03	0.695
13	2.21	3.29
14	1.03	3.6
15	1.4	3.6
16	1.65	3.27
17	2.98	2.88
18	3.77	1.28
19	2.6	1.317
20	2.41	1.57
AVG:	2.33635	1.59495
STD:	1.36915	1.050889

Table 5 TASK: PUT BLOCK-1 ON TABLE

NO. OF RUNS	HUMAN TIME	ROBOT TIME
1	1.9	0.965
2	1.28	0.949
3	0.74	4.32
4	3.12	3.6
5	2.11	3.6
6	4.2	2.97
7	3.03	2.97
8	1.55	1.46
9	0.865	1.09
10	2.61	1.39
11	2.99	1.02
12	1.81	0.89
13	2.88	0.898
14	2.98	1.24
15	1.05	1.21
16	1.25	1.00
17	0.563	1.05
18	2.15	0.964
19	1.39	0.843
20	2.13	0.88
AVG:	2.0299	1.66545
STD:	0.94395	1.095235

Table 6 TASK: PUT BLOCK-2 ON BLOCK-1

NO. OF RUNS	HUMAN TIME	ROBOT TIME
1	4.42	0.65
2	0.97	0.819
3	0.527	3.45
4	3.09	3.6
5	2.21	3.6
6	1.8	2.91
7	2.26	2.87
8	0.145	1.29
9	2.65	1.74
10	1.83	1.89
11	1.99	2.16
12	0.288	0.99
13	0.91	0.841
14	0.849	1.06
15	2.03	1.34
16	0.786	0.933
17	0.648	0.951
18	1.88	1.07
19	0.306	0.897
20	1.31	0.845
AVG:	1.54495	1.6953
STD:	1.055569	1.002854

Table 7 TASK: PUT BLOCK-3 ON BLOCK-2

NO. OF RUNS	HUMAN TIME	ROBOT TIME
1	1.92	1.69
2	0.419	0.876
3	0.394	3.48
4	3.37	3.6
5	3.43	3.6
6	3.57	3.22
7	2.68	2.85
8	2.55	0.986
9	0.593	1.57
10	0.13	1.519
11	0.659	0.645
12	2.31	0.873
13	0.469	0.774
14	1.88	0.87
15	1.02	0.915
16	1.8	1.00
17	2.82	0.901
18	1.42	0.875
19	3.11	1.02
20	1.93	1.05
AVG:	1.8087	1.6157
STD:	1.100917	1.04334

Table 8: ASSIGNMENT FREQUENCY TABLE FOR TASK: REMOVE BLOCK-2 FROM BLOCK-1

NO. OF RUNS	HUMAN FREQUENCY	ROBOT FREQUENCY
1	0	1
2	0	1
3	1	0
4	1	0
5	0	1
6	0	1
7	0	1
8	0	1
9	1	0
10	1	0
11	0	1
12	0	1
13	0	1
14	0	1
15	0	1
16	0	1
17	0	1
18	0	1
19	0	1
20	0	1
TOTAL:	4	16

NOTE: 1 - Assigned to resource with the smallest time

Table 9: ASSIGNMENT FREQUENCY TABLE FOR TASK: PUT BLOCK-1 ON TABLE

NO. OF RUNS	HUMAN FREQUENCY	ROBOT FREQUENCY
1	0	1
2	1	1
3	0	0
4	0	0
5	1	0
6	0	1
7	1	1
8	0	1
9	0	0
10	0	1
11	0	1
12	0	0
13	1	1
14	1	1
15	1	0
16	0	1
17	0	0
18	0	1
19	1	1
20	0	1
TOTAL:	7	13

NOTE: 1 - ASSIGNED TO RESOURCE WITH THE SMALLEST TIME.

Table 10: ASSIGNMENT FREQUENCY TABLE FOR TASK: PUT BLOCK-2 ON BLOCK-1

NO. OF RUNS	HUMAN FREQUENCY	ROBOT FREQUENCY
1	0	0
2	0	0
3	1	1
4	1	0
5	1	1
6	1	0
7	1	0
8	1	1
9	0	0
10	1	1
11	1	1
12	1	1
13	0	0
14	1	0
15	0	0
16	1	0
17	1	0
18	0	0
19	1	1
20	0	0
TOTAL:	13	7

NOTE: 1 - ASSIGNED TO RESOURCE WITH THE SMALLEST TIME.

Table 11: ASSIGNMENT FREQUENCY TABLE FOR TASK: PUT BLOCK-3 ON BLOCK-2

NO. OF RUNS	HUMAN FREQUENCY	ROBOT FREQUENCY
1	0	1
2	1	0
3	0	1
4	1	0
5	1	0
6	0	1
7	0	1
8	1	0
9	1	0
10	1	0
11	0	1
12	0	1
13	1	0
14	0	1
15	0	1
16	0	1
17	0	1
18	0	1
19	0	1
20	0	1
TOTAL:	7	13

NOTE: 1 - ASSIGNED TO RESOURCE WITH THE SMALLEST TIME.

FIGURE 14: GRAPH OF TASK TIME
(Remove Block-2 from Block-1)

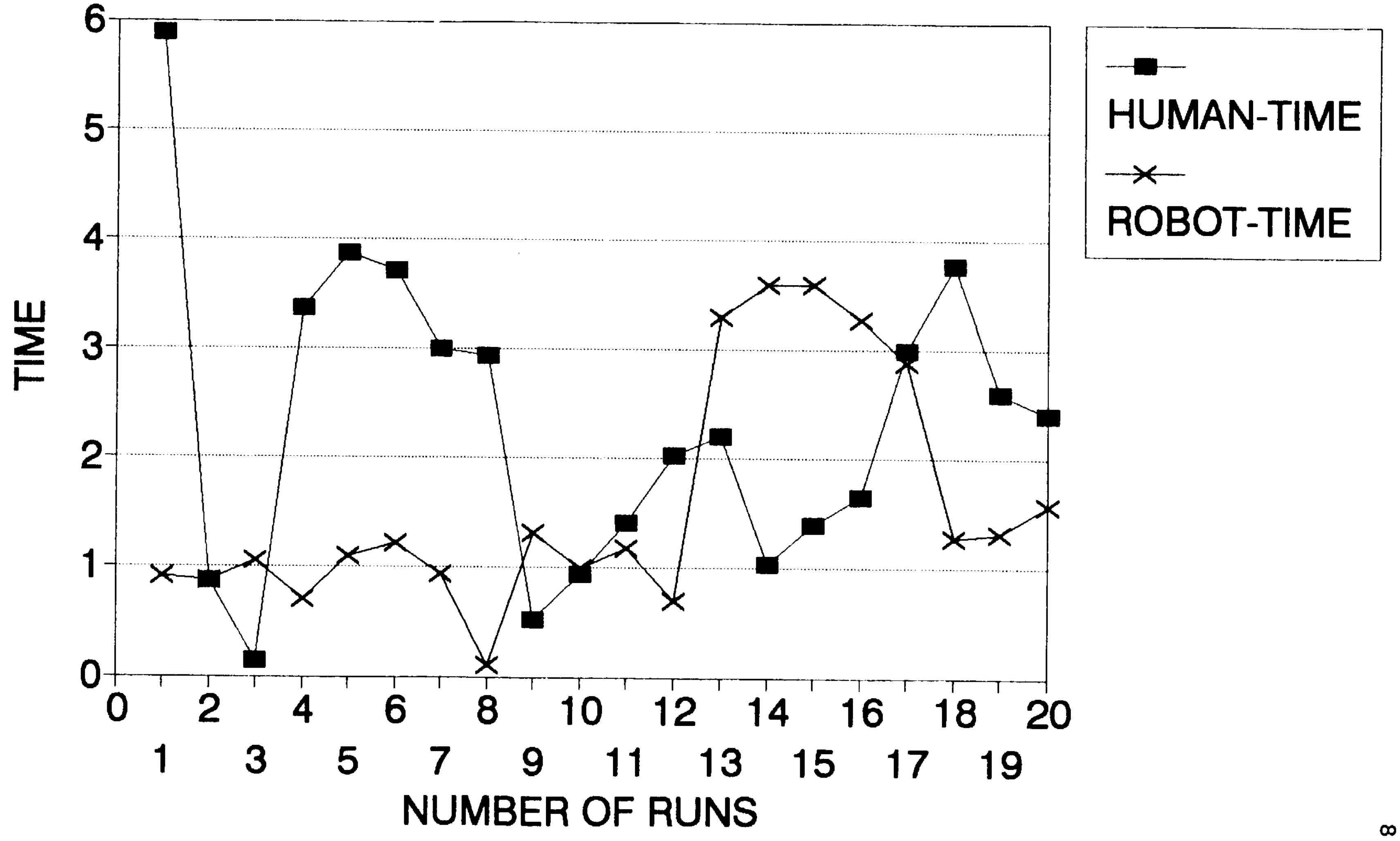


FIGURE 15: GRAPH OF TASK TIME
(Put Block-1 on Table)

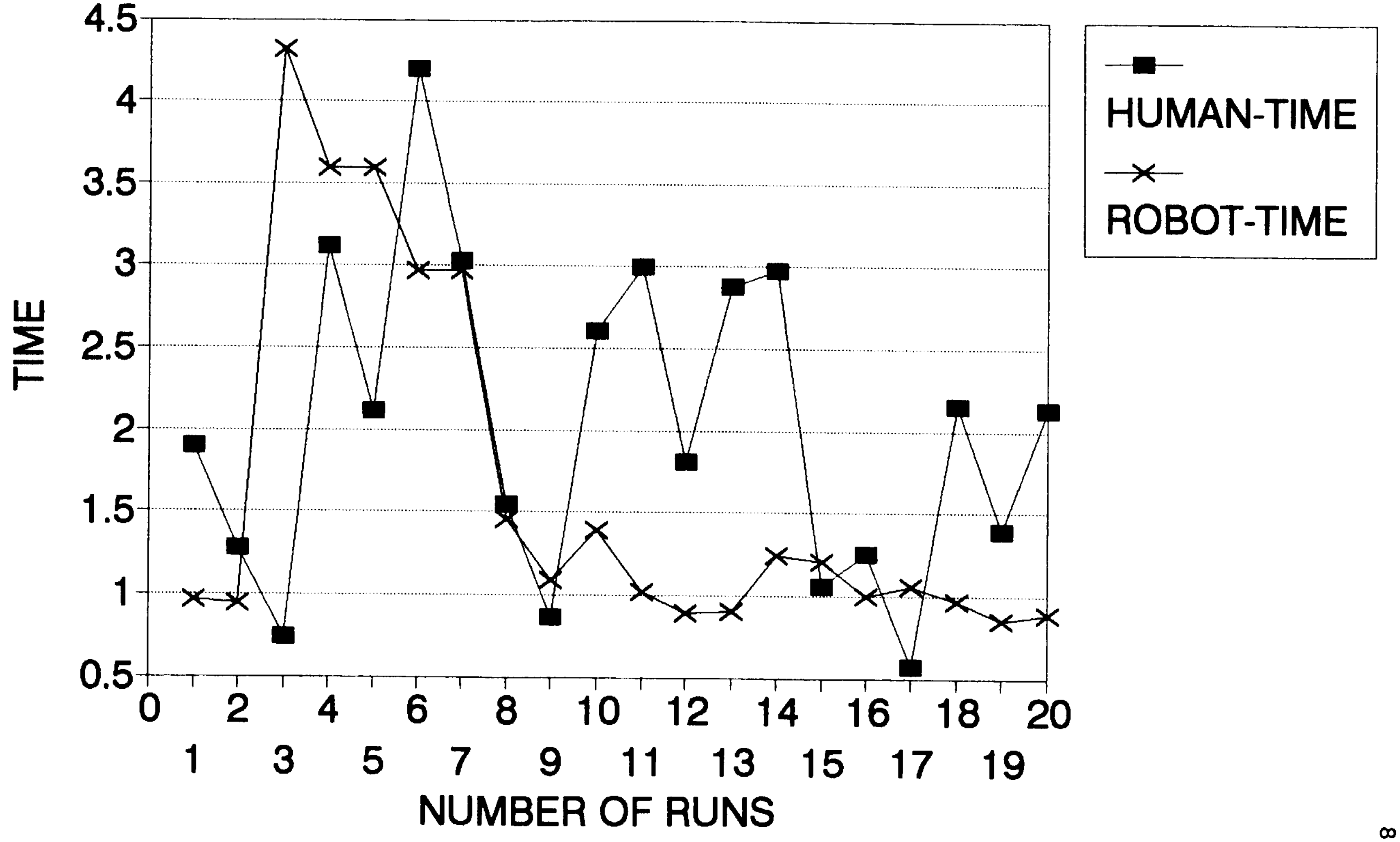


FIGURE 16: GRAPH OF TASK TIME
(Put Block-2 on Block-1)

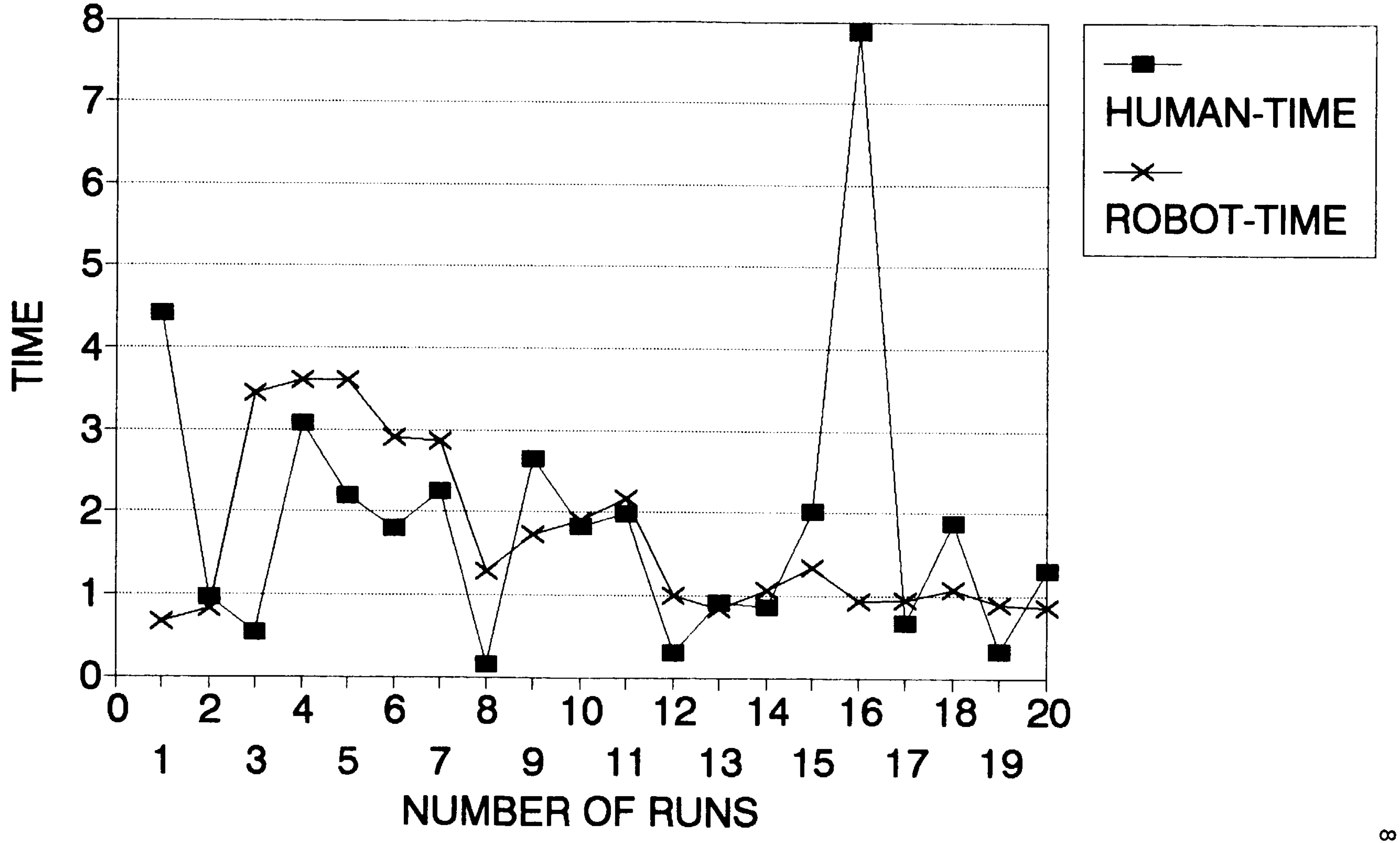


FIGURE 17: GRAPH OF TASK TIME
(Put Block-3 on Block-2)

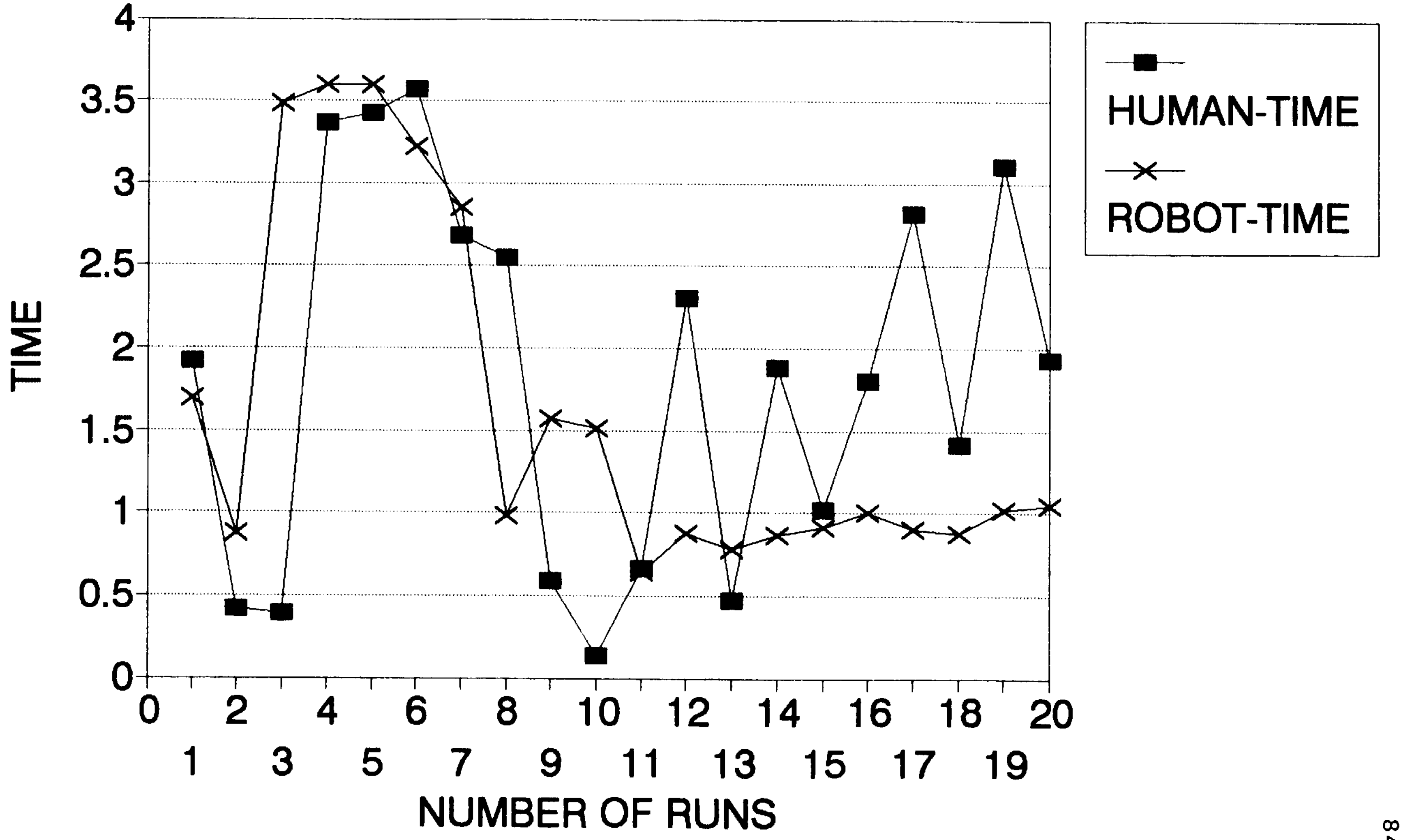


FIG. 18: FREQUENCY PLOT FOR ASSIGNMENT PROFILE (REMOVE BLOCK-2 FROM BLOCK-1)

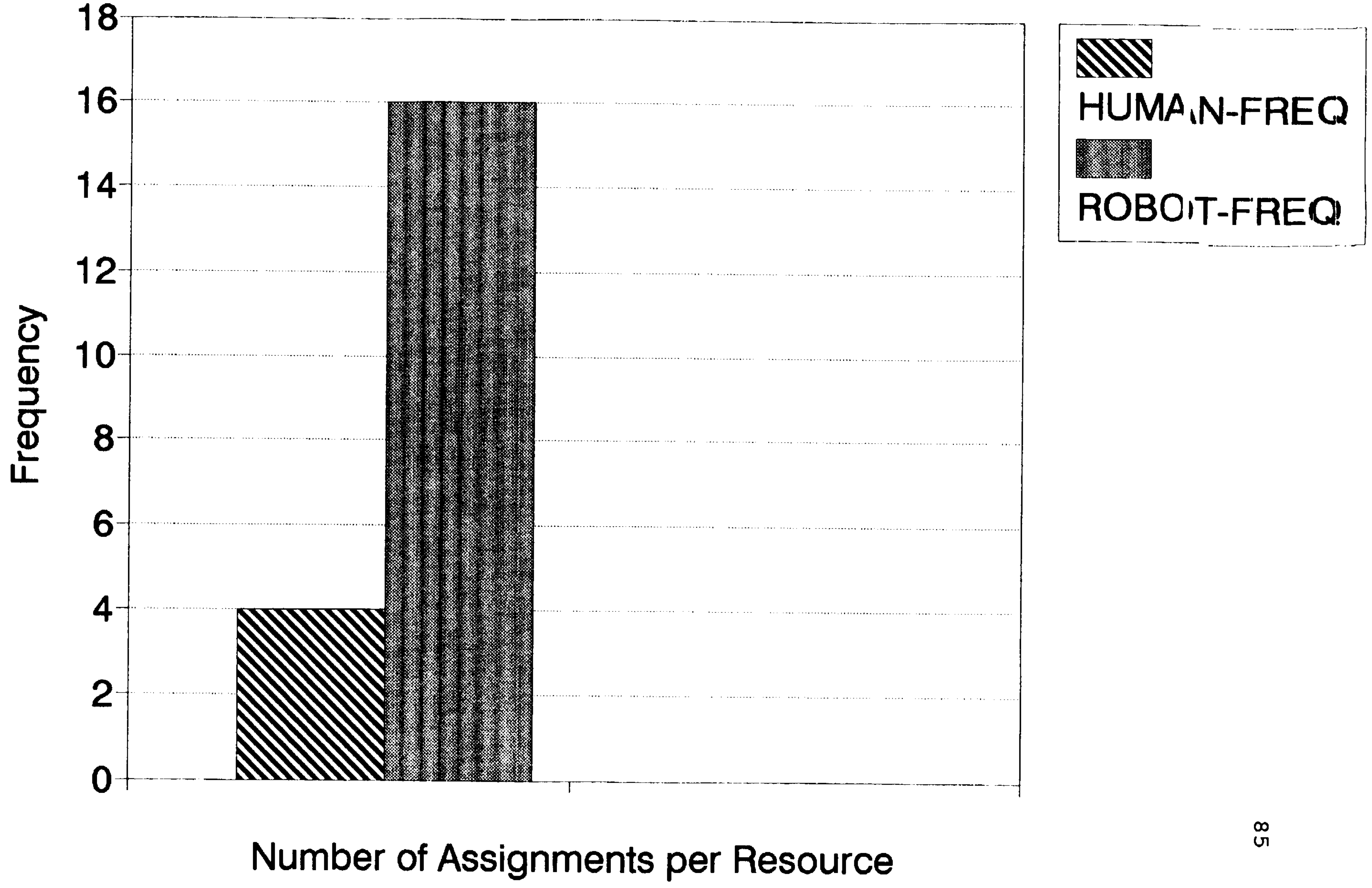


FIG. 19: FREQUENCY PLOT FOR ASSIGNMENT PROFILE (PUT BLOCK-1 ON TABLE)

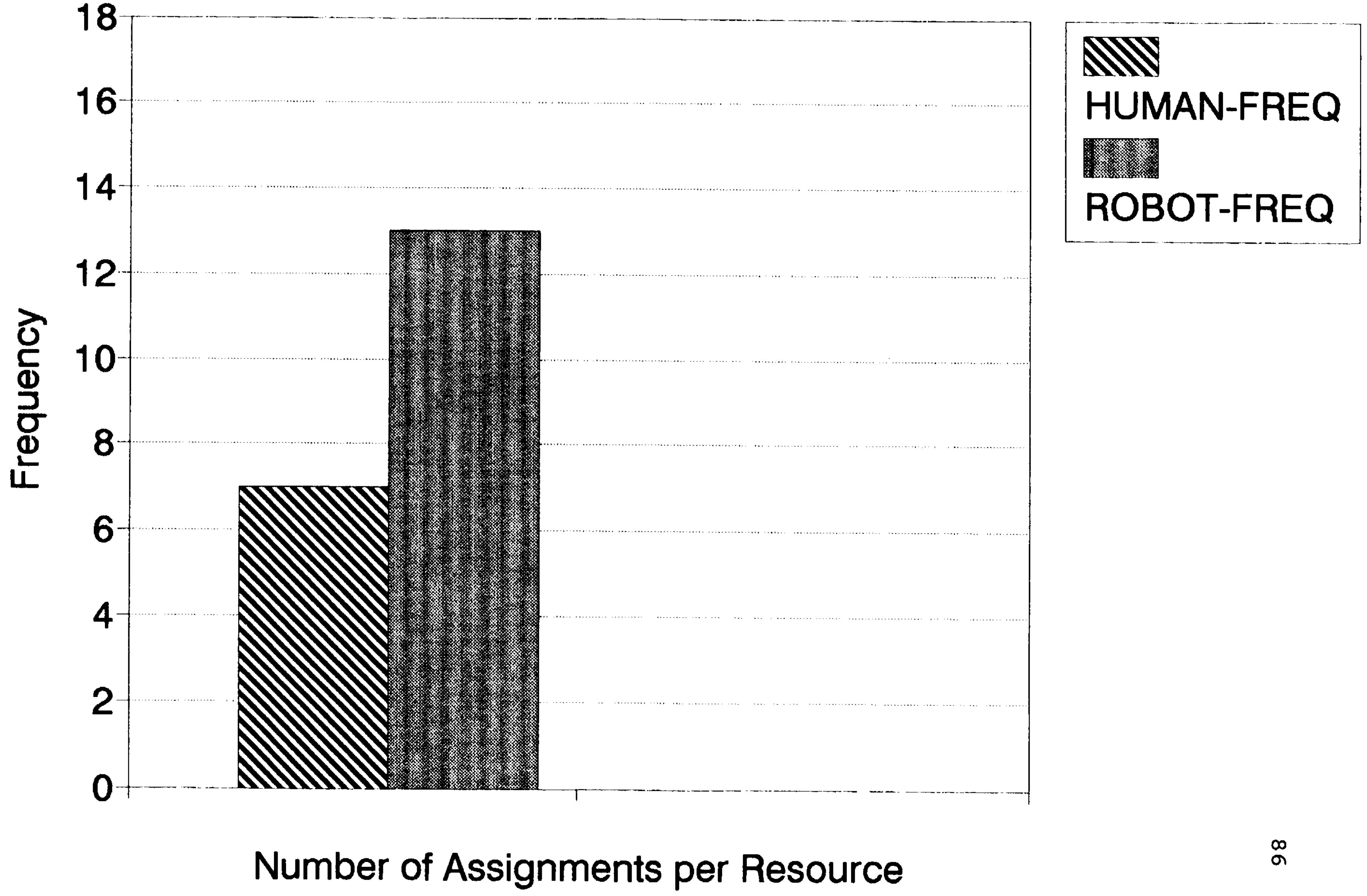


FIG. 20: FREQUENCY PLOT FOR ASSIGNMENT PROFILE (PUT BLOCK-2 ON BLOCK-1)

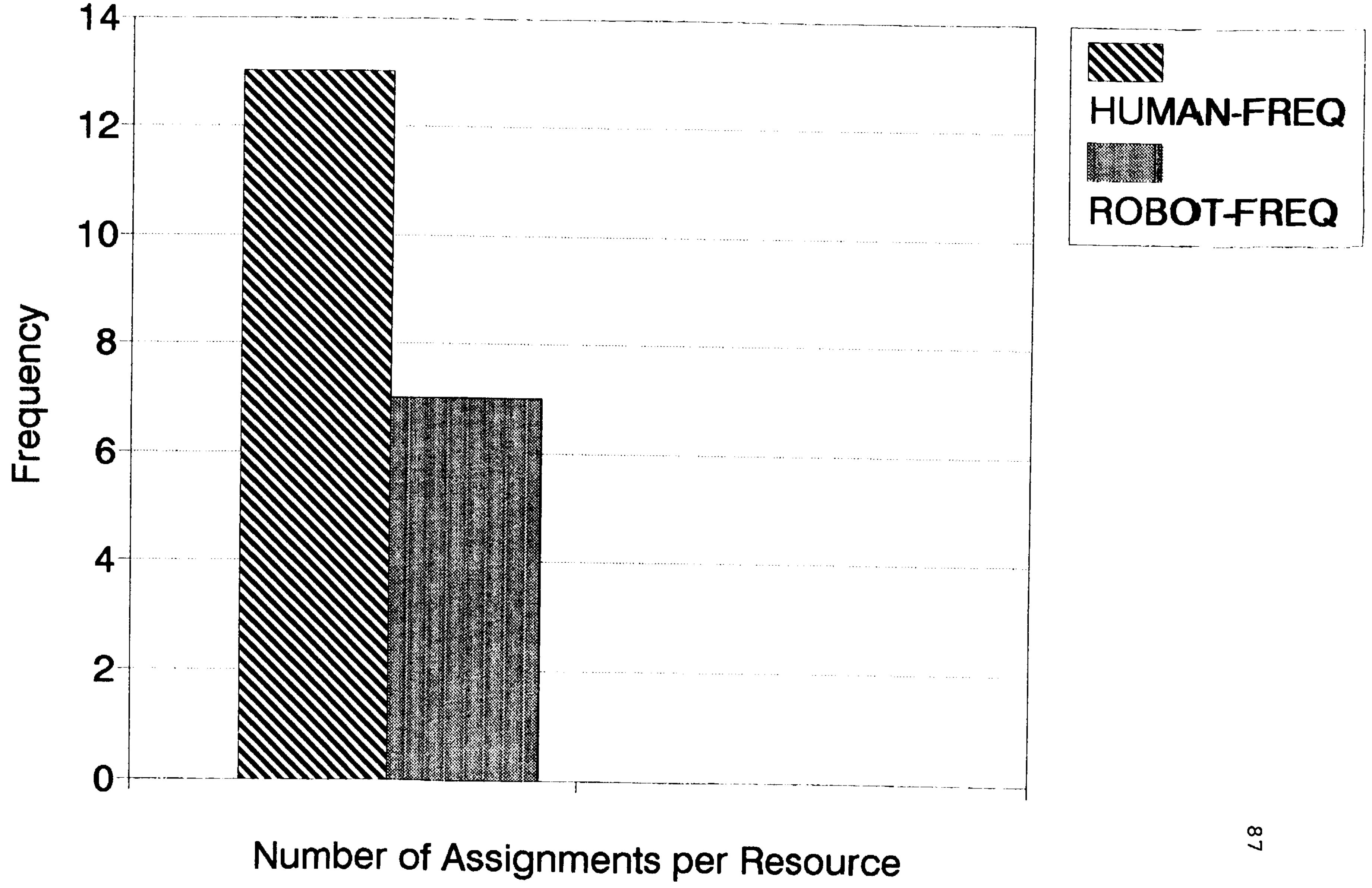
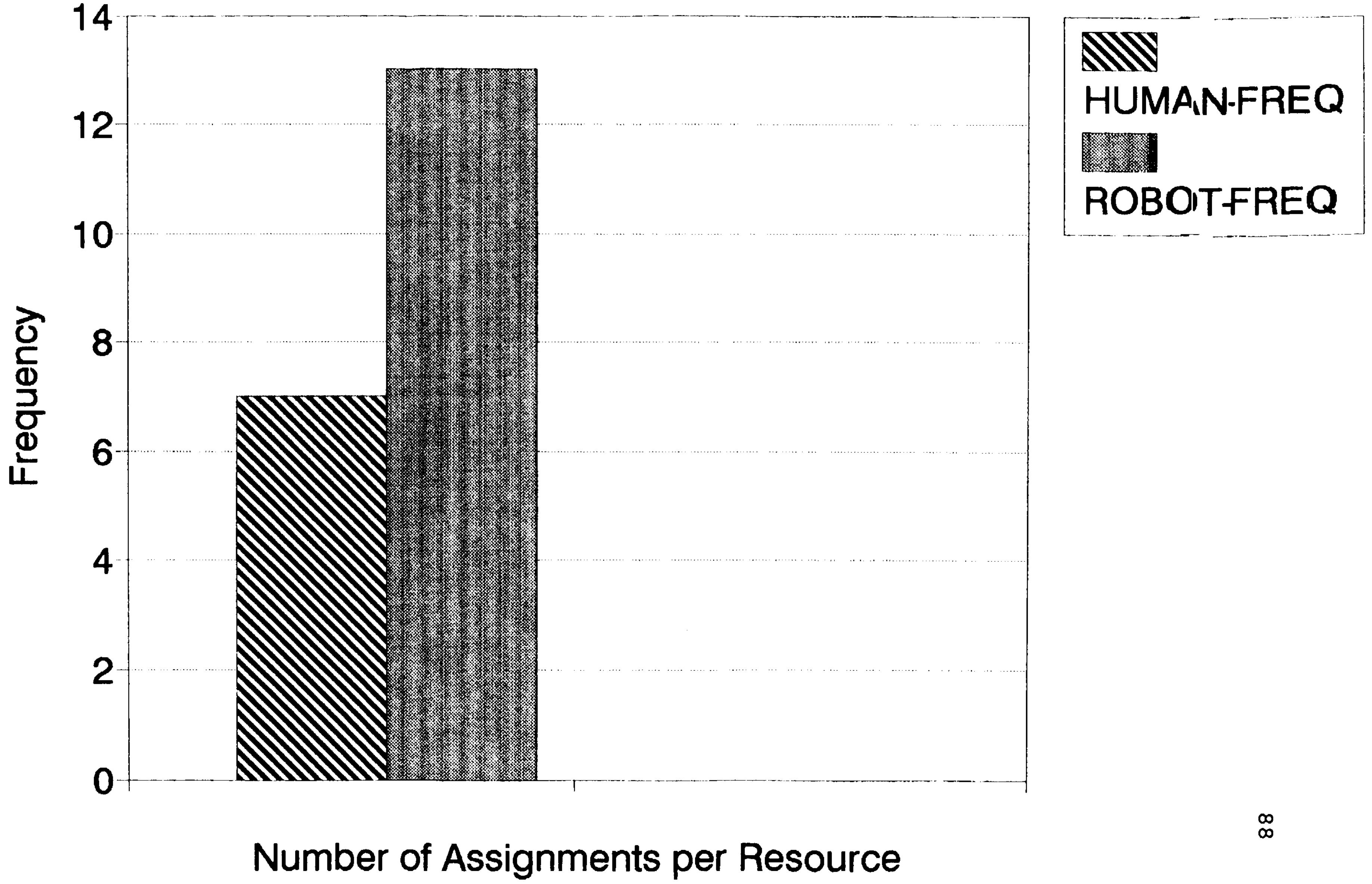


FIG. 21: FREQUENCY PLOT FOR ASSIGNMENT PROFILE (PUT BLOCK-3 ON BLOCK-2)



6.4 SUMMARY

In this chapter, we have presented two applications of TOP. The results obtained in the aircraft turnaround function prove to be more practical than the microblock problem. It was the experience gained from the microblock problem that led to the use of expert system shell (NEXPERT™[4]) in TOP. The microblock world problem however uncovers some issues dealing with teleoperation. Among these are:

- Repetitive tasks can be assigned to the robot while the human becomes the supervisor instead of a controller.
- Tasks involving high risk (such as nuclear waste handling) need to be taught to the robot. Thus, it is imperative that methodologies for robot acquisition of human skills be investigated.

The application of TOP to the aircraft turnaround function specifically addresses the issue of scheduling in an unstructured environment. The use of rule-based algorithms allow the system to be domain-independent. Thus, with a proper environment definition (data base preformating) the TOP can respond to such a new environment without a possible knowledge "degeneration." With the ability to reconfigure and replan, scheduling in TOP heuristically avoids the so called non solvable (NP hard) problems which are classically inherent in well defined job-shop or flow-shop systems.

[4] NEXPERT™ is a treadmark of Neuron Data, Inc.

CHAPTER 7

PROJECT SUMMARY

7.1 Accomplishment

Planning and scheduling in a telerobot system are two complex tasks because of the human-machine requirements that must be addressed. In a human system, these issues can well be approached from behavioral models. On the other hand, robot systems can be planned algorithmically by exploiting the available computational techniques.

A teleoperation requires direct cooperation between the agents involved in the system. Thus, a methodology to achieve such a cooperation must be developed. In this project, we have developed a computational approach to assigning tasks to multiagents working cooperatively in jobs that require a telerobot in the loop. Of course, our approach allows for such concepts to be applied in a non-teleoperational domain. We enumerate our accomplishment as follows:

1. In achieving a human-machine planning that co-exists based on the system characteristics, we have developed a planner that exploits human behavior during problem solving as well as subsuming robotic control models. The planning tool box, known as a "Deliberate Plan Network (DPN)" uses the principle of flow mechanism to preserve certain reasoning characteristics. In the DPN, decision elements try to evaluate each agent as a decision maker participating in a problem-solving task. The "best" decision is made using a

recursive algorithm to explore the DPN for the goal state. The programmability of DPN is transparent and easy for replanning. This is so because the fuzzy-based task skill-matrix can be evaluated on line and nodes or branches in DPN not satisfying the most eminent task assignment criterion pruned, re-evaluated, and/or updated.

2. Considering the fact that a telerobot operates in a hostile and non-structured environment, task scheduling should respond to environmental changes. In this regard, we have developed a general heuristic for scheduling jobs in a human-machine symbiotic system. Our technique is not to optimize a given scheduling criterion as in classical job -- and/or flow -- shop problems. For a teleoperation job schedule, criteria are situation dependent. A criterion selection is fuzzily embedded in the task-skill matrix computation. However, we have emphasized goal achievement with minimum expected risk to the human operator.
3. Our results with microblock world simulation experiments show that certain repetitive tasks done by human operators are easily to be taught to the robot. In such cases, the robot becomes the task controller and executer while the human becomes the job supervisor. The experiments further reveal that in certain (constraint) job environments, especially where human risk is very high, completion or task execution time is not as important as long as the robot can do the job with no risk to the human in the loop.

7.2 Further Work

Although the concept of teleoperation is not new, the use of telerobot in a more "intelligent" fashion needs a lot of research. The planning and scheduling discussions we have presented represent a subset of several works in the area of telerobotics. If we agree to look at a telerobot as a human-machine system, then the following fundamental questions are posted as the premises of the basic research issues of the future:

1. **INTENTIONAL PLANNING**: How can a plan understand the agent's intention?

   Problem Statement: In problem-solving systems, the use of objectives, intentions, and goals are often confused and/or used interchangeably. Therefore, in building a planner for a telerobotic environment, these terms show up as constraints. A methodology for aggregating such constraints and resolving their behavioral conflicts must be addressed.

2. **KNOWLEDGE ORGANIZATION**: a) What type of organized knowledge is needed for the process of deciding what to do in a multiple agent environment? b) How can knowledge in control, command, communication and intelligence (C^3I) be planned to function modularly in an unstructured environment with multiple telerobots at the same time supporting their symbiotic roles?

   Problem Statement: A telerobotic system requires a multifarious knowledge structure. At the conceptual level, there is the problem of multiple abstraction and the representation of knowledge. At the contextual level, there is problem of commonality; that is, the degree to which an aspect of a system is common to other parts of the whole system. This, in a classical sense, can be referred to as a "symbiotic anomaly". At the experimental level, there is the problem of recognizing the individual agent problem-solving strategy and articulating these into a rule set which can simulate the actual scenario. We must resolve these problems in developing a useful planner for a telerobot-human environment.

3. **CONSTRAINT DIRECTED PLANNING**: How do we handle constraints and resolve conflicts in planning a limited resource system?
Problem Statement: Human action is marked by a striking flexibility that can not be predetermined by a plan. However, in a telerobot system, a common goal is to be realized by aggregating and solving problems with multiple goal functions. Usually, the resources required to realize such goals may be limited. Various questions to be answered during planning should attempt to address the basis of allocating tasks to agents. In particular, how and on what basis should a task be performed conjunctively?, independently?, or collectively?

4. **REPLANNING AND PLAN DIAGNOSIS**: How can a plan reconfigure its knowledge base and its strategies in an unexpected situation and for contingency?
Problem Statement: A telerobot domain may require addressing at least two faulty conditions: a) anomaly; where an actual event is so alien to the planner's expectations that it falls

entirely outside the bounds of the contingency set; b) conjunctivity; when an actual event may require the cooperation of at least two robots. Thus, any instance where a planning system is confronted by some situation not covered by any predefined course of action may require the need of some sort of adaptive planning. The intent is to deal with anamolous situations that might arise unexpectedly during life cycle of the system. The question is, can a distributive contingency system with self diagnostic capability be built into a planner with multiple goals?

5. **HUMAN-MACHINE INTERACTION**: If plans are defined as "virtual resources" in situated actions, how are the capabilities of the human and the robot characterized? What kind of information processing paradigm is required in a virtually heterogeneous multi-agent systems?

Problem Statement: A telerobotic work environment requires an information processing paradigm different from the classical off-line programming languages used for industrial robots. Currently, most researchers address this issue from a functional approach. Typically, the problem of explicit versus implicit communication dominates most literary discussions. For a telerobotic system, an "intelligent" human-machine interaction concept must be pursued. For example: a) how do we relate actual human psychology to information (computing) psychology, both from the user and system level perspective? b) how do we match human physiology (ergonomic factors) and mechanical physiology (machine hardware) in planning a task? c) how are human knowledge, skills, and rules related to software/hardware integration for real-time planning? d) since interaction is a style of control and is accomplished through a language medium, how can a planning knowledge be codified in a domain-specific style while preserving generality for applications?

REFERENCES

1. Action, W., Perez, W., and Reid, G. (1986). "On The Dimensionality of Subjective Workload". Proceedings of the Human Factors Society 30th Annual Meeting, Dayton, Ohio, September 29-October 3, Vol. 1, pp. 76-80.

2. Albus, J.S., Barbera, A.J., and Nagel, R.N. (1981); "Theory and Practice of Hierarchical Control". Proceedings of the Twenty-Third IEEE Computer Society International Conference.

3. Antsaklis, P.J., Passino, K.M., and Wang, S.J. (1989). "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues:. Journal of Intelligent and Robotic Systems, Vol. 1 (4), pp. 315-3342.

4. Atkinson, D.J. (1988). "Telerobot Task Planning and Reasoning: Introduction to JPL AI Research". Proceedings of Space Telerobotics Workshop, pp. 339-350.

5. Carbonell, J.R. (1969). "On Man-Computer Interaction: A Model and Some Related Issues". IEEE Transactions on Systems Science and Cybernetics SSC-5 (No. 1), 16-26.

6. Chern, M.Y. (1984). "An Efficient Scheme for Monitoring Sensory Conditions in Robot Systems". Proceedings of the IEEE International Conference on Robotics, Atlanta, GA pp. 298-303.

7. Chu, Y.Y., and Rouse, W.B. (1979). "Adaptive Allocation of Decision Making Responsibility Between Human and Computer in Multi-Task Situation". IEEE Transactions on Systems, Man, Cybernetics SMC-9 (No. 12), 769-778.

8. Coiffet, P. (1981). Robot Technology, Vol. 1: Modeling and Control. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

9. Davis, P.R. (1977). Using and Re-using Partial Plans. Ph.D. thesis, University of Illinois at Urbana-Champaign.

10. Decker, K.S. (1987). "Distributed Problem-Solving Techniques: A Survey," IEEE Trans. on Systems, Man, and Cybernetics, SMC-17 (5), September/October, pp. 729-740.

11. Dorfman, P.W., and Goldstein, I.L. (1971). "Spatial and Temporal Information As Cues In A Time-Sharing Task". Journal of Applied Psychology, Vol. 55, pp. 554-558.

12. Dunker, O. et al (1988). "The Effectiveness of Supervisory Control Strategies in Scheduling Flexible Manufacturing System". IEEE Transactions On Systems, Man, Cybernetics, SMC-18(2)., pp. 228-237.

13. Enstorm, K.D., and Rouse, W.B., 1977. "Real-Time Determination of How a Human Has Allocated His Attention Between Control and Monitoring Tasks." IEEE Transactions on Systems, Man, and Cybernetics. MC-7 (No. 3), pp. 153-161.

14. Fahlman, S.E. (1974). "A Planning System for Robot Construction Tasks". Artificial Intelligence, 5(1), pp. 1-49.

15. Ferrell, W.R., and Sheridan, T.B. (1967). Supervisory Control of Remote Manipulation. IEEE Spectrum 4(10), 81-88.

16. Fikes, R.E., and Nilsson, N.J. (1972). "Some New Directions In Robot Problem Solving". In Machine Intelligence (B. Mectzer & D. Michie, Eds.), Vol. 7, pp. 405-430.

17. , Hart, P.E. and Nilsson, N.J. (1972). "Executing Generalized Robot Plans", Artificial Intelligence, 3(4), pp. 251-288.

18. , and Nilsson, N.J. (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". Artificial Intelligence, 2, pp. 189-208.

19. Fox, B.R., and Kempf, K.G. (1985). "Opportunistic Scheduling for Robotic Assembly". IEEE Intl. Conf. on Robotics and Automation, St. Louis, MO., pp. 880-889.

20. Fox, M.S. (1983). "Constraint Directed Search: A Case Study of Job-Shop Scheduling". Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA.

21. Gasser, L. and Bekey, G. (1987). "Task Allocation Among Multiple Intelligent Robots". Proceedings of the 1987 Workshop on Space Telerobotics, Vol. 1 (Rodriquez, G., ed.), JPL, Pasadena, CA., pp. 127-130.

22. Govindaraj, J. (1979). "Modeling the Human as a Controller in a Multi-Task Environment", Ph.D. thesis, University of Illinois at Urbana-Champaign.

23. Govindaraj, T., and Rouse, W.B. (1979). "Modeling Human Decision Making in Multi-Task Situations Involving Both Control and Discrete Task? Proceedings of the Fifteenth Annual Conference on Manual Control, Wright State University. Wright Patterson AFB, OH: Air Force Flight Dynamics Laboratory.

24. Greenstein, J.S. (1979). "Human Decision Making in Multi-Task Situations: Event Detection, Attention Allocation, and Implications for Computer Aiding". Ph.D. thesis, University of Illinois at Urbana-Champaign.

25. Harmon, S.Y. (1988). "Dynamic Task Allocation and Execution Monitoring in Teams of Cooperating Humans and Robots. Proceedings Workshop on Human-Machine Symbiotic Systems, Oak Ridge Tennessee, December 5-6; pp. 79-98.

26. Harrison, F.W., and Orlando, N.E. (1984). "System-Level Approach Automation". Fourth Annual UAH/UAB Robotics Conference, Huntsville, Alabama, April 26.

27. Hayes-Roth, B., and Hayes-Roth, F. (1979). "Cognitive Model of Planning". Cognitive Science, Vol 3, pp. 275-310.

28. Hirai, S., and Sato, T. (1985). "Advanced Master-Salve. Manipulator:. The 15th International Symposium on Industrial Robots, Tokyo Japan, pp. 137-144.

29. Jenkins, L. (1988). "Space Telerobotic Systems: Applications and Concepts". Proceedings of Space Telerobotics Workshop, pp. 29-34.

30. Knowles, W.B. (1963). "Operator Loading Tasks", Human Factors, Vol. 5, pp. 155-161.

31. Koivo, A.J. and Bekey, G.A. (1988). "Report of the Workshop on Coordinated Multiple Robot Manipulators: Planning, Control and Applications". IEEE Journal of Robots and Automation, RA-4 (1), pp. 91-93.

32. Kok, J.J., and Van Wink, R.A. (1977). A Model of the Human Supervisor. Proceedings of the Thirteenth Annual Conference on Manual Control". Moffett Field, CA: NASA, pp. 210-216.

33. Kuipers, B. (1977). "Modeling Spatial Knowledge". 5th Intl. Conference on Artificial Intelligence, MIT, Cambridge, MA, pp. 292-298.

34. Lesser, V.R. and Corkill, D.D. (1981). "Functionally-Accurate, Cooperative Distributed Systems," IEEE Trans. on Systems, Man, and Cybernetics, SMC-11 (1), pp. 81-96.

35. Licklider, J.C.R. (1960). Man-Computer Symbiosis. IEEE Transactions on Human Factors in Electronics HFE-1 (No. 1), 4-11.

36. Malone, T.B., Kirkpatrick, M., and Kopp, W.H. (1986). "Human Factors Impact on System Workload and Manning Levels". Proceedings of the Human Factors Society, 30th Annual meeting, pp. 763-767.

37. Martin, H.L., and Kuban, D.P. (1985). Teleoperated Robotics In Hostile Environments. Dearborn: SME Publications.

38. McCallan, M.I. and Reid, L. (1982). "Plan Creation, Plan Execution and Knowledge Acquisition in a Dynamic Microworld". Intl. Journal of Man-Machine Studies.

39. Newell, A.G., Shaw, J.C. and Simon, H.A. (1960). "Report on a General Problem Solving Program". Proc. Intl. Conference on Information Processing, UNESCO, Paris, pp. 256-264.

40. , and Simon, H.A. (1972). Human Problem Solving. Prentice-Hall, Englewood Cliffs, New Jersey.

41. Nof, Sy.S., Knight, J.L., Jr., and Salvendy, G. (1980). "Effective Utilization of Industrial Robots: A Job and Skills Analysis Approach", AIIE Transactions, Vol. 12(3), 216-225.

42. Norcoss, R.J. (1988). "A Control Structure for Multi-tasking Workstations." Proceedings, of the IEEE International Conference on Robots and Automation, Philadelphia, pp. 1133-1135.

43. Ntuen, C.A. and Park, E.H. (1991). "An Experiment in Human-Robot Interaction During Task Execution". Proc. of IEEE Intl. Conf. on Systems, Man and Cybernetics. (to appear).

44. Ntuen, C.A. and Park, E.H. (1988). "A Fuzzy Expert Simulation For Dynamic Task Allocation in A Telerobotic Environment," The Proc. of 1988 Southeastern Simulation Conference, Oct. 17-18, Orlando, Florida, 122-126.

45. Ntuen, C.A. and Park, E.H. (1989). "Artificial Intelligence Planning Systems: A Review and Analysis For Industrial Engineering Applications." Proceedings of the International Industrial Engineering Conference, May 14-17, Toronto, Canada.

46. Ntuen, C.A. and Park, E.H. (1988). "Intelligent Teleoperation And Man-Machine System Simulations. The Proceedings of 1988 Southeastern Simulation Conference, October 17-18, Orlando Florida, pp. 118-121.

47. Ntuen, C.A., Park, E.H. and Sliwa, N.O. (1988). "Knowledge Requirements for Modeling A Telerobotic Environment as a Man-Machine System". Proceedings of Institute of Industrial Engineers Conference. Orlando, FL. May 22-25), 138-143.

48. Orlando, N.E. (1983). "A System For Intelligent Teleoperation Research", AIAA Computers in Aerospace Conf., Hartford, CN.

49. Orlando, N.E. (1984). "An Intelligent Robotic Control Scheme", Presented at the American Control Conf., San Diego, CA.

50. Parker, L.E. and Pin, F.G. (1987), "Dynamic Task Allocation for a Man-Machine Symbiotic Systems", Report # ORNC/TM-10397/CESAR-87/08, Oak Ridge National Lab.

51. Pearson, G. (1988). "Mission Planning For Autonomous System." Proceedings Space Telerobotics Workshop, pp. 303-306.

52. Roach, J., and Boaz, M. (1985). "Coordinating the Motions of Robot Arms In a Common Workspace". Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, MO, pp. 494-499.

53. Robinson, A. (1983). "Research in Planning". SIGART Newsletter, January, pp. 27-36.

54. Rouse, W.B. (1973). A Model of the Human in a Cognitive Prediction Task. IEEE Transactions on Systems, Man, and Cybernetics SMC-3 (No. 5), 473-477.

55. Sarcerdoti, E.D. (1975). "Sturcture For Plans and Behavior". Ph.D. thesis Stanford University CA.

56. __________(1974). "Planning in a Hierarchy of Abstraction". Artificial Intelligence, 5, pp. 115-135.

57. Saridis, G.N., (1983). "Intelligent Robot Control". IEEE Transactions Automatic Control. AC-28, pp. 547-556.

58. Sato, T., Hirai, S. (1987). "Language-Aided Robotic Teleoperation System (LARTS) for Advanced Teleoperation", IEEE Journal of Robotics and Automation, RA-3(5), 476-481.

59. Sato, T., Hirai, S. (1978). "Motion Understanding and Structured DD Master-Slave Manipulator For Cooperative Teleoperator". Third Intl. Conference on Advanced Robotics (ICAR' 87), October 13-15, Versailor, France.

60. Sheridan, T.B. (1988). "Man-Machine Communication for Symbiotic Control". Workshop on Human-Machine Symbiotic Systems proc. (Parker, L.E. and C.R. Weisbin, Eds.), Oak Ridge, Tenn., pp. 21-35.

61. Sheridan, T.B. (1989). "Telerobotics". Automatics, 25(4), pp. 487-507.

62. Shin, K.G. and Epstein, M.E. (1987). "Intertask Communications In An Integrated Multirobot System". RA-3 (2), pp. 90-100.

63. Silverman, B.G., "Distributed Inference and Fusion Algorithms for Real-Time Supervisory Controller Positions," IEEE Trans. on System, Man, and Cybernetics, SMC-17(2) March/April 1987, pp. 230-239.

64. Smith, R.G. and Davis, R. (1981). "Frameworks For Cooperation In Distributed Problem Solving". IEEE Transactions on Systems, Man, and Cybernetics, SMC-11 (1), pp. 61-70.

65. Stank, L. (1988). "Telerobotics: Research Needs Evolving Space Stations". Proceedings Space Telerobotics Workshop, pp. 91-94.

66. Stefik, M.J. (1981). "Planning and Meta-Planning". Artificial Intelligence, 16(2), pp. 141-169.

67. (1981). "Planning With Constraints". Artificial Intelligence, 16(2) pp. 11-140.

68. Swartout, W. (1988). "DARPA Santa Cruz Workshop on Planning". AI Magazine, Summer, pp. 115-130.

69. Tangwongsan, S. and Fu, K.S. (1979). "An Application of Learning to Robot Planning". Intl. Jnl. of Computer and Info. Science, (4), pp. 626-650.

70. Tate, A. (1977). "Generating Project Networks". Proc. of the 5th Intl. Conf. on AI, MIT, Cambridge, MA., pp. 888-893.

71. Vere, S.A. (1983). "Planning in Time, Windows and Duration for Activities and Goals". Pattern Analysis and Machine Intelligence PAMI-5(3), pp. 246-266.

72. Yang, J-Y. D. et. al (1985). "An Architecture For Control And Communications In Distributed Artificial Intelligence Systems." IEEE Transactions on Systems, Man, and Cybernetics, SMC-15 (3), pp. 316-326.

73. Wilensky, D.E. (1981). "Meta-Planning". Cognitive Sciences, 5, pp. 197-233.

74. Zadeh, L.A. (1973). "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes". IEEE Transactions on Systems, Man, and Cybernetics SMC-3 (No. 1), 28-44.

APPENDIX

DPN - A Deliberate Plan Network in TOP

The theory of deliberation is that humans typically have a purpose in mind when they attempt to execute a defined action. We use this concept to generalize a case of where multiple agents (not necessarily humans) have to interact cooperatively during task execution. In particular, we extend the "deliberate behavior" to the robotic agents. The robot is taught the same action primitives through a deliberate plan network. The description of a DPN follows.

The initial development of a DPN is achieved via context definition blocks as shown in Fig. A-1. From the task definition block, the planner takes in the initial task status (similar to STRIPS concept of initial operator state) with the goal definition to develop plans. Constraints are constructively posted based on resource availabilities. The deliberator elements (also known as bureaucratic fixers) are used to compare the plan goal and the expected constraints to be encountered. Upon deliberation, the human agent would use the moderator elements (conflict resolution program) to relax constraints and perform task assignments. The operators are similar in context to those used by STRIPS and NOAH.

The task definition block describes the composition of tasks to be performed. The block "structural task planner" describes the structural relationship between the tasks. The constraint block describes the order in which the tasks are to be performed and the resources required. The "Schedule Activities" block is the program that converts plans into time phase schedules. The external blocks

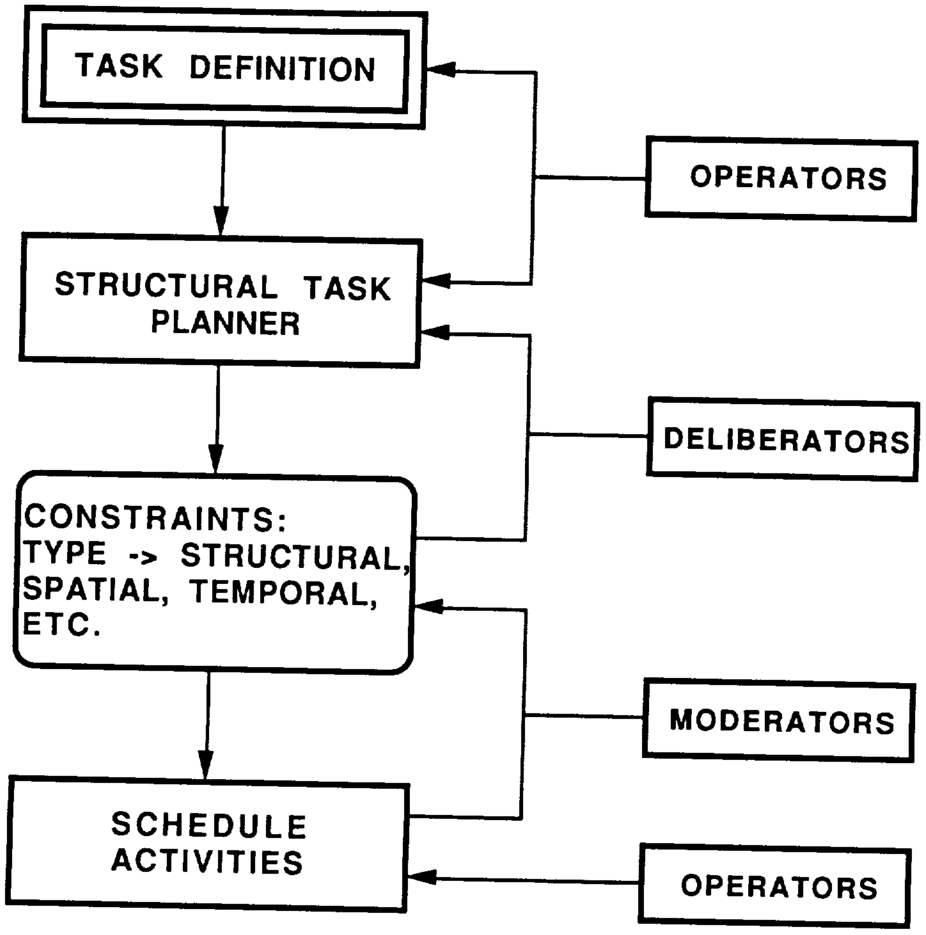


Fig. A-1: Context Definition Blocks for a DPN.

are "Operators", "Deliberators", and "Moderators". The operators are the problem solvers or plan executors. For example, the operator

MOVE(object,fposition,nposition)

moves the object from a current position (fposition) to a new position (nposition) in the desired task configuration space. The "Deliberators" use some weighted performance measures from the teleoperator skill matrix to construct task assignments. The "moderators" are programs for constraint resolutions and relaxations.

An example of plan deliberation during a task execution is illustrated in Fig A-2. The "attempt task execution" block labelled "A" defines the goal and the initial state of the system. The "verify predicate node" B is a decision node which checks for prerequisite conditions prior to attempting task execution. If the verify - predicate is true, the current subgoal achievement is protected (by flagging) and put into the "agenda" knowledge base. From then on, the operator designated to execute the subtask (E block) is searched and the subtask execution takes place at block L. A return condition to the verify-predicate of block B is to check for completion of the subtasks required for the goal achievement.

Should the verify-predicate condition become false, the moderators and deliberators are used to resolve potential constraints (block G). Any conflict in task assignment is checked in block H. A wrong task assignment indicates a subgoal failure

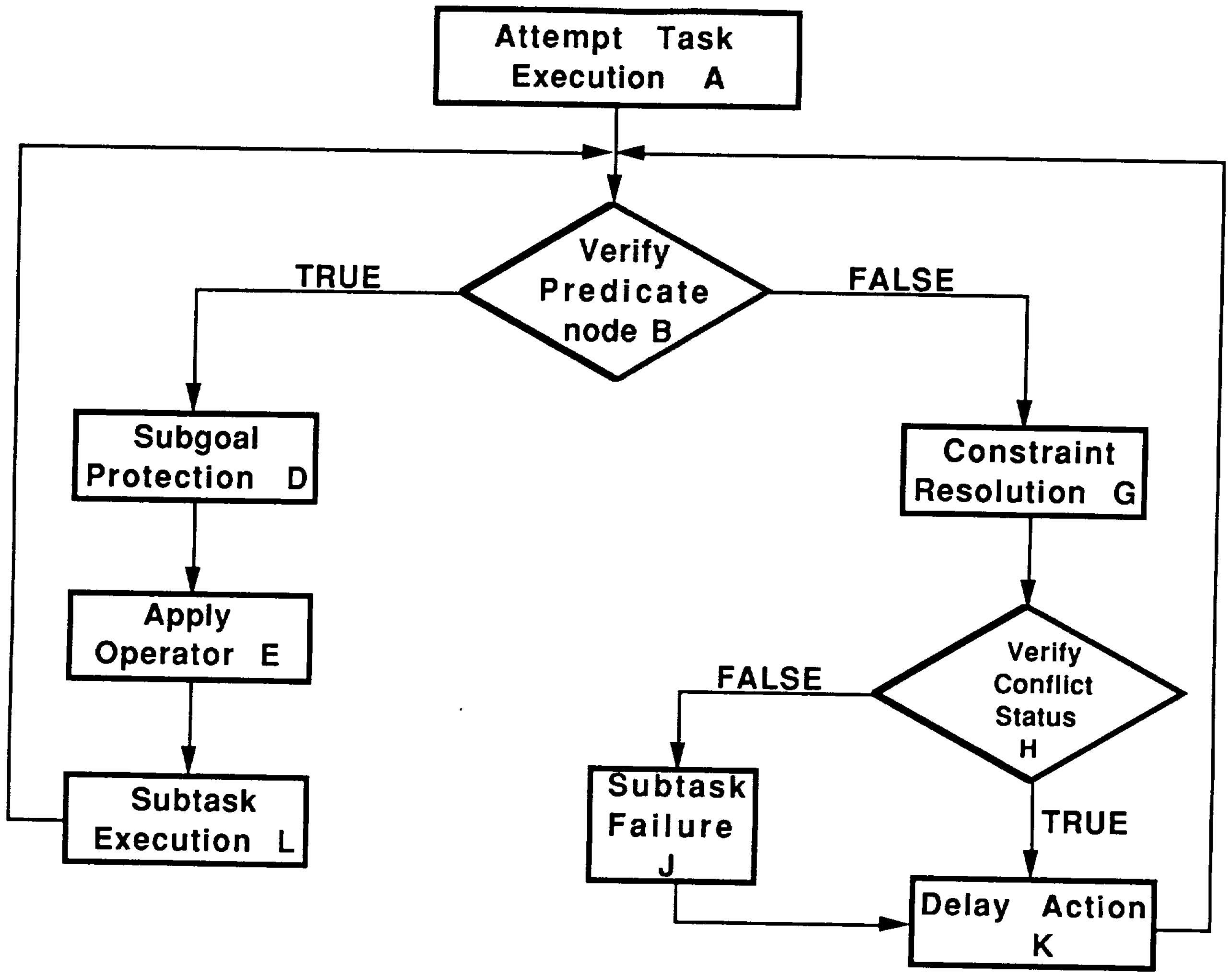


Fig. A-2: An Example of Plan Deliberation during Task execution.

(block J), and a no feasible assignment is sent directly to the "delay-action" block K from where the planner looks back to the verify-predicate block B for other feasible subtasks to be executed.

The representation of the plan mechanism above is shown as a reduced skeleton state space network in Fig. A-3. As shown, the operators and deliberators are active in some nodes. The technique is to remove all moderators and deliberators and preserve operators for task execution without failure. The planning heuristic is as follows:

1. Eliminate loops where moderators and/or deliberators are active.
2. Apply constraint relaxation algorithms on nodes with deliberators.
3. Repeat steps 1-2 until only task execution operators are active.
4. Attempt task execution.
5. If subtask failures are encountered, redefine the task scenario and go to step 2.
6. Else, protect successful subtasks and put them in task agenda.
7. If the agenda contains all the subtasks for a goal achievement, schedule task execution.
8. Else, define a new subtask scenario. Go to step 1.

The above discussion is shown in Figs. A-4 and A-5 respectively. An example MULISP syntax for operator, deliberator and moderator is as follows:

```
(Defun  GOAL ('put-on (a &path b)): goal definition

  (Attempt(&path a b)) :      operator to attempt
                              putting block a on b
```

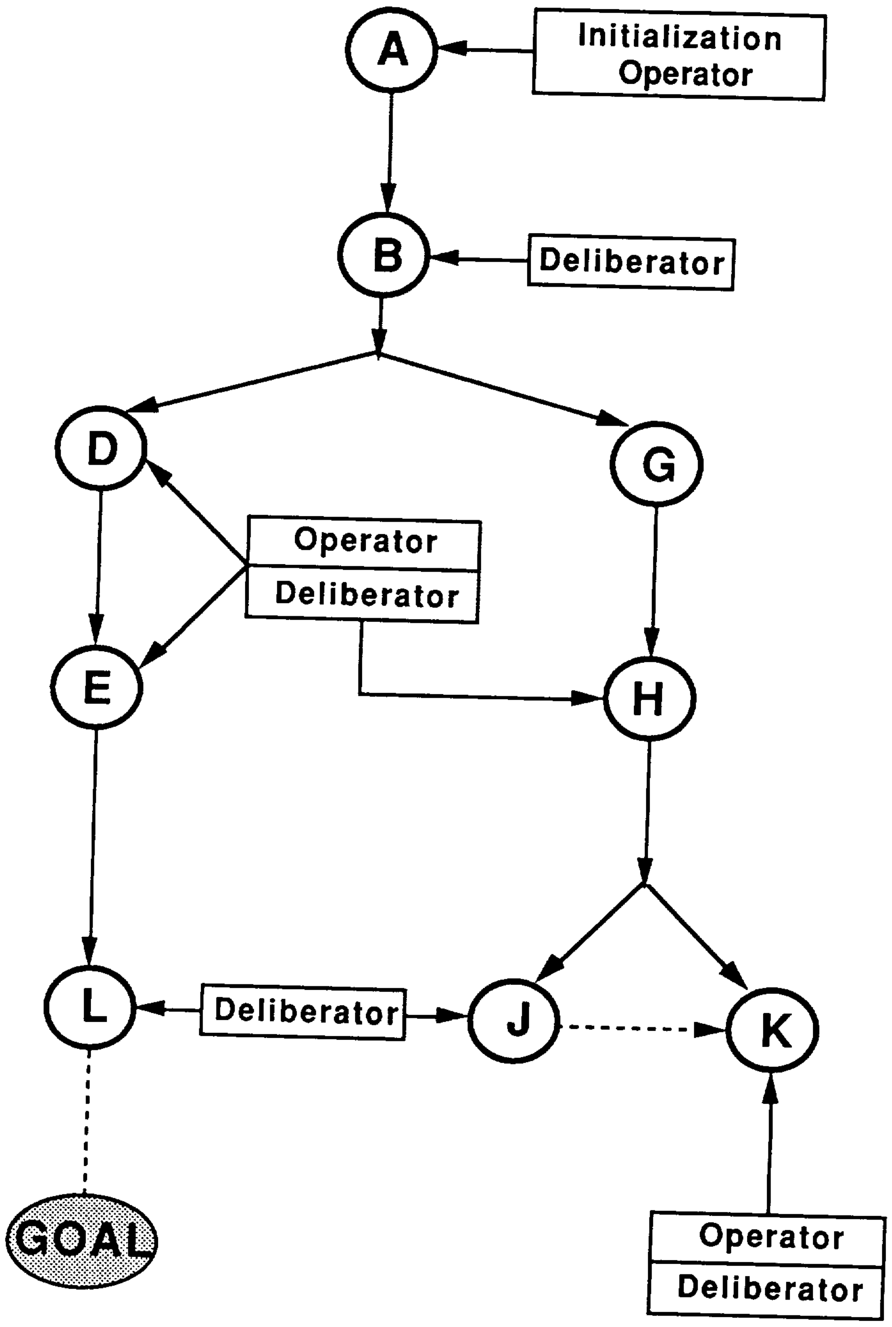


Fig. A-3: A Reduced State Space Deliberate Plan Network.

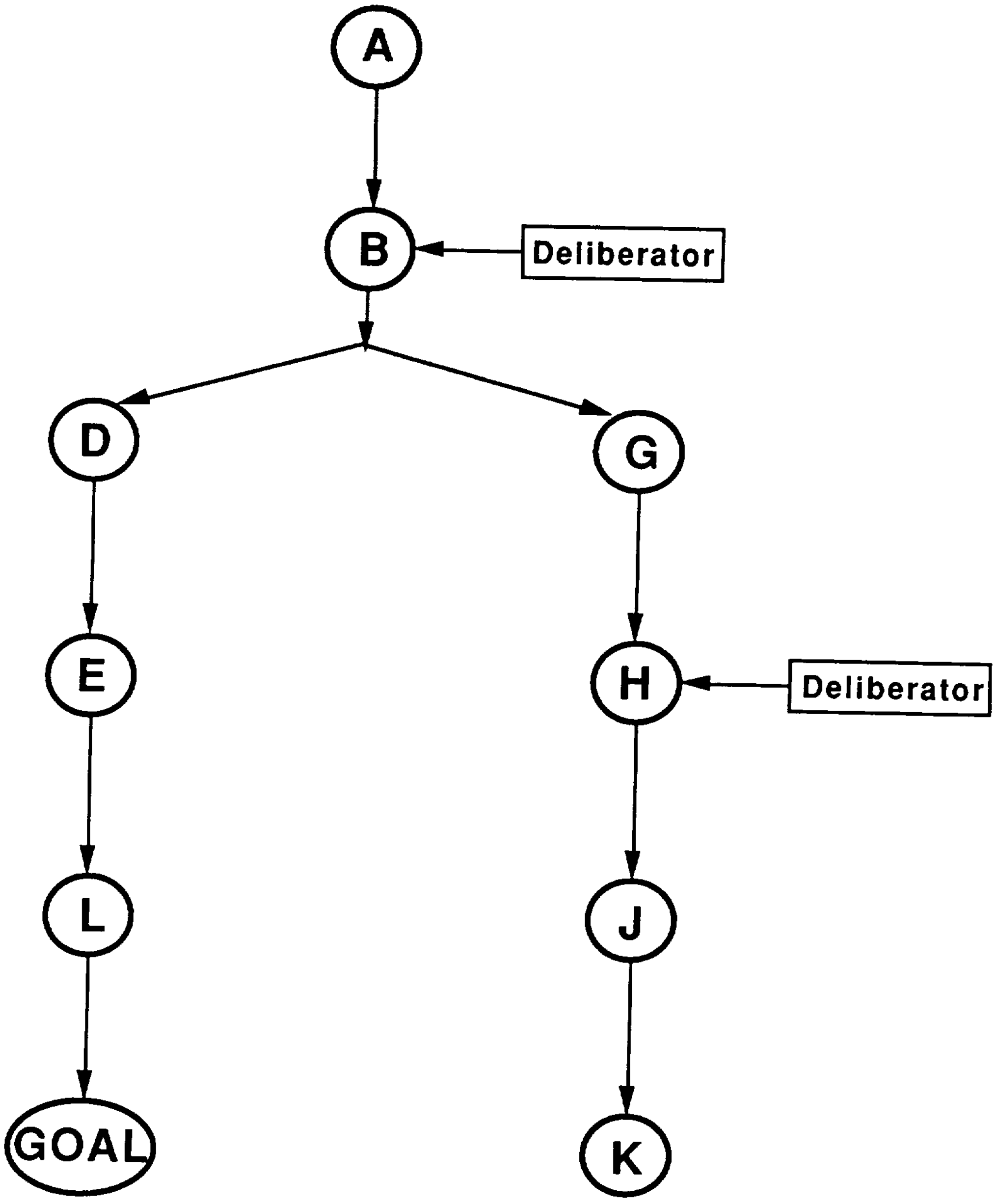


Fig. A-4: An Example DPN with Incomplete Goal Attainment due to Existence of False Predicates in Nodes B and H.

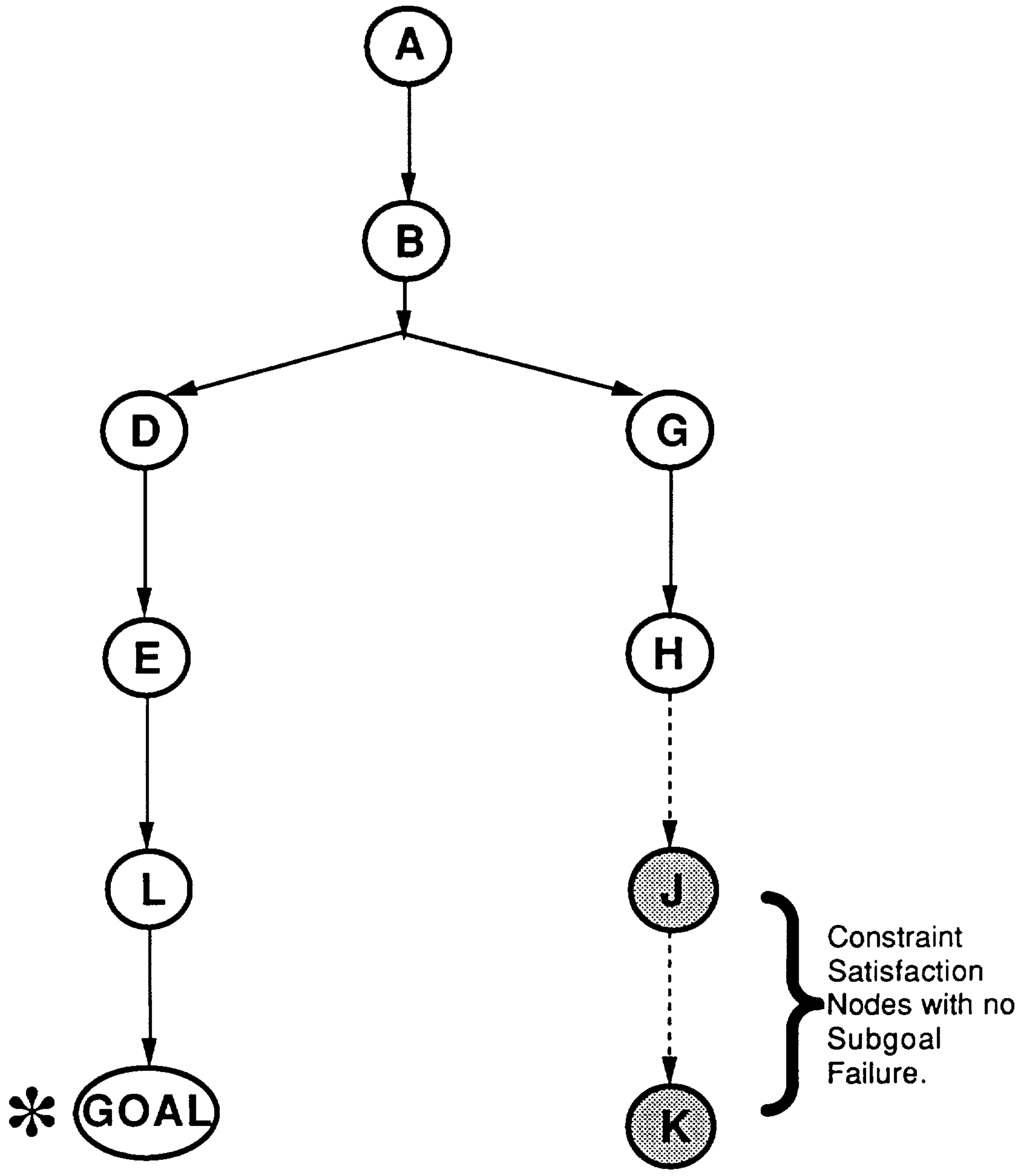


Fig. A-5: An Example Solution Graph With No Subgoal Failure With All Constraints Satisfied.

```
            (if (verify-p, status),:  verify current state
                  fail                configuration,
              (protect status) ))) :  failure, repeat protect
                                      subgoal.
```

The DPN plan expansion pragmas is based on lambda procedure as follows:

```
     (LAMBDA(X Y)
  (Apply(#'Attempt(null x y ))
       #' Expand (x y) ))
```