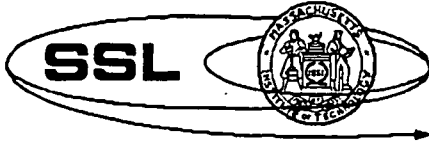


GODDARD-GRANT

919.



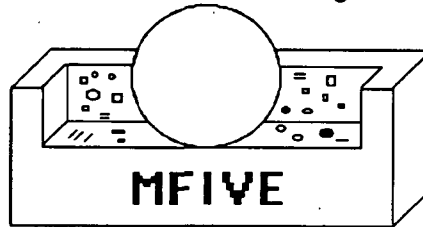
IN-16763

AS MJ 700 802

Final Report

July 31, 1986

Study of Onboard Expert Systems to Augment Space Shuttle and Space Station Autonomy



N86-25886

(NASA-CR-176958) STUDY OF ONEBOARD EXPERT SYSTEMS TO AUGMENT SPACE SHUTTLE AND SPACE STATION AUTONOMY Final Report, 1 Aug. 1984 - 31 Dec. 1985. (Massachusetts Inst. of Tech.) 91 p

Unclas
CSCL 22B G3/18 43242

Grant NAG5-445

Submitted to

NASA Goddard Space Flight Center
Code 402, Greenbelt, MD 20771

From the
Space Systems Laboratory
Massachusetts Institute Of Technology
77 Massachusetts Ave., Room 33-407
Cambridge, Massachusetts 02139

SSL Report #16-86

Contributors:

Clifford R. Kurtzman
David L. Akin
David Kranzler
Edith Erlanson

TABLE OF CONTENTS

<u>Section</u>	<u>Page Number</u>
Section 1: Introduction	1
Section 2: Crew Activity Planning	5
2.1 Division of Scheduling Functions Between Space and Ground Crews	5
2.2 An Overview of Planning and Scheduling: Artificial Intelligence Techniques	8
2.3 An Overview of Planning and Scheduling: Operations Research Techniques	10
2.4 Space Station Scheduling Model	12
Section 3: Stowage Logistics	15
3.1 The Stowage Logistics Clerk	15
3.2 Stowage Management System Specifications	22
3.3 Expert System Concepts	22
3.4 Expert System Design	24
3.5 The Design of the Stowage Logistics Clerk	25
Section 4: The MFIVE Crew Scheduler/Logistics Clerk ---	29
4.1 Development Goals	29
4.2 MFIVE Operation	30
4.2.1 The Data Base Management System	32
The Crewmember Information Form	32
The Job Information Form	38
The Flight Plan Information Form	41
The Tool Information Form	44
The Stowage Information Form	50
The Options Menu	50

TABLE OF CONTENTS

<u>Section</u>	<u>Page Number</u>
4.2.2 The Scheduler	53
The Scheduling Worksheet	53
Selecting a Job to be Scheduled	59
Jobs Requiring Multiple Crewmembers	62
Time Constraints	65
Target Constraints	70
Other Options	70
4.2.3 The Tool Searcher	73
Section 5: Conclusions and Recommendations	77
Appendix A: An Optimal Algorithm for the Propagation of Time Constraints	81
A.1 Problem Overview	81
A.2 Constraint Reduction	82
A.3 Inference Generation	83
A.4 Algorithms	83
A.5 Implementation Considerations	85
A.6 Storage Considerations	87
Bibliography	89

Section 1: Introduction

Activities onboard a space station may be grouped into two categories: core activities (those required to keep the station operating and the crew in good health) and mission activities (those relating to performing the tasks for which the station is required, such as satellite servicing, EVA structural assembly, materials processing and manufacturing, and technology experiments). While both types of tasks are necessary, they may be thought of as competing for the limited resources of the space station crew. Any savings in time or effort required to complete the core tasks would then become available to mission-oriented activities.

Skylab showed that these core activities could occupy a large amount of the crew's time. In the weightless environment, storing and retrieving unused pieces of hardware were both difficult and time-consuming. Of equal importance to system performance is the impact of core activities on the required number of ground personnel. Between keeping track of where equipment was stowed and planning the crew activities for the following day, a sizable work force was required for Skylab at JSC Mission Control; a similar situation has also developed for Space Shuttle operations. In addition, the activities of the Skylab ground stowage monitoring procedures were often ineffective, as crews failed to communicate information on hardware stowage to the ground. This mode of operation will significantly increase the operating cost of a space station. Soviet experience has also indicated difficulty in finding stowed equipment [Ivakhnov, 1984].

In addition, there is evidence that the psychological health of a space station crew would be enhanced by giving them the ability to have some control over their schedules, as would be provided by an interactive scheduling system. Scheduling done from the the ground, without crewmember input, has often resulted in "overcontrolling" of the crew, with deleterious consequences. An example of this occurred during the last Skylab mission in December 1973. According to the flight director at the time: "We send up about six feet of instructions to the astronauts' teleprinter every day, at least 42 separate instructions telling them where to point the solar telescope and which scientific instruments to use. We lay out the whole day for them, and they normally follow it to a T. We've learned how to maximize what you can get out of a man in one day" [Balbaky, 1980]. After over a month in space, the three astronauts conducted the first "strike in space" by turning off the radio and taking a day off [Cooper, 1976].

It has therefore been suggested that the utility and autonomy of space station operations could be greatly enhanced by the incorporation of computer systems utilizing expert decision making capabilities and a relational data base. An expert decision making capability will capture the expertise of many experts on various aspects of space station operations, for subsequent use by nonexperts (i.e., spacecraft crewmembers). Key features of an expert system are its ability to explain, on request, how it arrived at its decisions, and the capability for users to reject the system's conclusions if they disagree with them. The information utilized by expert systems is stored in a relational data base that is used by the system in response to the current problem: the expert system uses rules to generate conclusions based on the information stored in the relational data base.

During the summer of 1983, NASA sponsored a summer workshop which investigated the potential ways in which machine technology could potentially affect and augment space station operation. One of the conclusions of that study, of which one of the MIT study team (Kurtzman) was a participant, was that crew activity planning was an attractive area for space station automation [Johnson, et al., 1985]. This study has therefore focused on upon demonstrating the feasibility of onboard crew activity planning. A secondary focus of the study has been the use of expert systems technology to aid crewmembers in locating stowed equipment. The two tasks are linked together; the use of tools and equipment is associated with the performance of onboard crew activities. It is envisioned that eventually these functions will be integrated into a unified space station computer system and relational data base, which could then support a wide variety of functions, such as those listed in Table 1.

Section 2 of this report discusses in detail the crew activity planning problem, along with a summary of past and current research efforts. The requirements and specifications which the study used to develop its crew activity planning system are also defined. Similarly, Section 3 of this report discusses the general stowage logistics problem, along with its requirements and specifications. The guidelines laid out in Sections 2 and 3 were used to create the MFIVE Crew Scheduler and Logistics Clerk, whose development and operation are discussed in Section 4. Section 5 presents the study's conclusions and recommendations, and Appendix A discusses a mathematical algorithm, used by the MFIVE Scheduler, which the study developed to aid in optimal crew activity planning.

TABLE 1

Potential Space Station Knowledge Base Applications (not an exhaustive list)

- 1 Subsystems Management Tasks
 - 1.1 Power Subsystem Management
 - 1.2 Thermal Subsystem Management
 - 1.3 Life Support Subsystem Management
 - 1.4 Information Subsystem Management
 - 1.5 Propulsion Subsystem Management
 - 1.6 Communications Subsystem Management
 - 1.7 Fluids Subsystem Management
 - 1.8 System Performance Evaluations/Trends Analysis
 - 1.9 Consumable Inventory and Resupply
 - 1.10 Fault Detection and Annunciation
 - 1.11 Troubleshooting/Malfunction Procedures
 - 1.12 In-Flight Maintenance

- 2 Crew Activity Planning
 - 2.1 Daily Housekeeping Chores
 - 2.1.1 Trash Removal
 - 2.1.2 General Cleaningetc.
 - 2.2 Maintenance and Repair Schedules
 - 2.3 Crewmember Health Maintenance
 - 2.4 Construction Activities
 - 2.5 Satellite Servicing Activities
 - 2.6 Scientific Experimentation Activities
 - 2.7 Scientific Observation Activities
 - 2.8 Training Activities
 - 2.9 Crew Rotation

- 3 Trajectory/Flight Dynamics
 - 3.1 Space Station Orbital Maintenance
 - 3.2 Orbiter Transfer Vehicle Tracking and Maneuver Planning
 - 3.3 Satellite Rendezvous Tracking and Maneuver Planning
 - 3.4 Orbiter Rendezvous Tracking and Maneuver Planning

- 4 Construction/Satellite Servicing
 - 4.1 Assemble Structures/Spacecraft
 - 4.2 Satellite Retrieval/Servicing
 - 4.3 Satellite Checkout

- 5 Long Term Planning/Logistics

Section 2: Crew Activity Planning

2.1 Division of Scheduling Functions Between Space and Ground Crews

The entire process of planning crewmember activities is a highly complex one involving consideration of and tradeoffs among crewmember workloads, resource usage, task priorities, and time and target constraints of various tasks to be performed by the crew. For shuttle missions, all planning and scheduling functions are currently done by ground schedulers. Due to the short nature of a shuttle mission and the very limited amounts of in-space crew time, it will probably be necessary to continue this practice, with any on-orbit scheduling activities limited at most to replanning in the event of unanticipated deviations from the nominal schedule. Aboard a space station, where people will be living for months at a time, it would be highly desirable for the crew to perform as much of the planning and scheduling process as is feasible. Due to the long duration of space station stays, it will be necessary to organize rational works and rest schedules which maintain crewmember health. Work cycles based on a 24 hour day may be standard, but allowance must be made for the occurrence of irregular or emergency situations which would require other work schedules [Litsov and Bulyko, 1983; Ivaknov, 1984]. Requiring the crew to entirely and autonomously manage the entire scheduling process, however, would require them to deal with vast amounts of data and perform functions which are beyond their areas of expertise; this would consume large amounts of valuable crewmember time. Ground controller input will still be necessary to assist the space crew, but in a much reduced role (both in terms of scope and manpower) over that necessary without a space based crew activity planning system. Table 2 indicates a functional allocation scheme for allocating crew scheduling activities between ground controllers and the space based crew.

Ground controllers must retain the ability to provide input to any space based activity planning system, but generally not in terms of short-term real-time control. Ground control will necessarily have strong input to overall mission priorities, and will need to provide the detailed technical data that will serve as computational parameters regarding requirements, resources, and constraints. This data will include such items as space station orbital parameters (e.g. inclination, apogee, perigee), task descriptions, time windows for performing the task, and instructions to assist the crew in performing the task. Ground operations will also undertake preliminary trial scheduling runs to help establish general mission feasibility and priorities.

PRECEDING PAGE BLANK NOT FILMED

Scheduling Activity or Task	Location of Activity or Task		
	Ground Operations	Expert System	Crew
Design of Expert System	✓		
Test and Demonstration of Expert System	✓		✓
Maintenance and Adaptation of Expert System	✓	✓ (long term)	✓
Assemble Input Data: Time Estimates, Resource Requirements, Constraints	✓		
Input Daily Task Priorities and Update Information	✓		✓ (secondary)
Preliminary Timeline Computation (Days in Advance)	✓ (monitoring)	✓	
Compute/Review Next-Day Timelines (Overnight)	✓ (monitoring)	✓	✓ (interactive)
Review/Update/Recompute Morning Timelines	✓ (monitoring)	✓	✓ (interactive)
Approve/Accept Final Morning Timelines	✓		✓
Graphically Represent Timelines		✓	
Recommend Deviations from Daily Timelines	✓	✓	✓
Contingent Recomputation of Daily Timelines		✓	✓

Table 2: Functional Allocation of Crew Scheduling Activities

[Adapted in part from Johnson, et al., 1985]

Onboard a space station, the scheduling system could then be operated in an interactive mode on a daily (or other routine) basis to determine crew work timelines. At the beginning of the work day, the crewmember in charge of scheduling functions could use the scheduling system to determine work timelines for that day. Activities could be interactively scheduled into time slots and assigned to crewmembers, with the scheduling system simply pointing out existing windows which satisfy all resource requirements and time constraints. This mode of operation affords the greatest flexibility to crewmembers in determining their work schedules. Subjective preferences by the crew may have minor effects on schedule quality, but large effects on crew morale. However, if all the activities which the crewmembers are to perform are scheduled in this manner, it would place an unacceptable burden on the crewmember performing the scheduling. In addition, it would not allow the utilization of mathematical techniques for scheduling optimization, which would maximize use of crewmember capabilities, utilization of the limited resources available, and achievement of mission goals.

A second mode of scheduler operation would allow the program to automatically plan out the remaining tasks. The scheduler would couple operations research algorithms and heuristics with the large computational capabilities of computers to generate optimal (or near optimal) timelines. Schedules could thus be determined in minutes which would have efficiencies comparable to ones requiring weeks or months of manpower to generate using traditional dispatching or trial and error approaches.

The scheduler would then provide crewmembers with printouts of timelines, instructions for performing the various tasks, and the locations of various tools. At the end of the work day, the scheduler could be briefed on the success or failure of each of the various tasks done that day (e.g., which items had to be rescheduled, which were not performed due to lack of time, etc.). If needed, the system could also be consulted during the work day for additional information and replanning.

A high priority in the development of a scheduler is to keep its operation as simple as possible, and to require as little crewmember training as possible to operate, while still permitting a large amount of crewmember input if desired. If the system is so cumbersome that it must be used constantly or require highly specialized training and skills to operate effectively, then the crewmembers will probably not want to use it at all.

2.2 An Overview of Planning and Scheduling: Artificial Intelligence Techniques

It is only within the past twenty years that artificial intelligence techniques have been available to create expert systems to assist in task planning. NOAH was an early planner which dealt with interacting subgoals. The method of least commitment and backward chaining initially produced a partial ordering of steps for each plan. When interference between subgoal plans was observed, the planner adjusted the ordering of the steps to resolve the interference and produce a final parallel plan with time ordered steps [Gevarter, 1982]. Other planning systems include STRIPS, which was developed to perform such tasks as stacking blocks to achieve a particular goal formation, and MOLGEN, which was designed to plan experiments in molecular genetics. The NUDGE system was developed to understand incomplete and possibly inconsistent management-scheduling requests and to provide a complete specification for a conventional scheduling algorithm. Perhaps one of the projects most closely related to crew activity planning is the ISIS expert system developed at Carnegie-Mellon University to perform job shop scheduling functions [Fox, et al., 1983; Smith, 1983; Townsend, 1983]. The OPAL expert system [Bensana, et al., 1986] and the MASCOT expert system [Erschler and Esquirol, 1986] are also being developed for aiding in job shop scheduling.

Over the past several years there has been great interest within NASA for using expert systems for space applications. One of the first systems which NASA developed was the Deviser system, built at JPL to autonomously plan a spacecraft's actions during a planetary flyby [Vere, 1983a; Vere, 1983b; Vere, 1985]. Deviser was developed from NOAH, but unlike NOAH, it is capable of handling goals with time constraints and durations. JPL is also developing a system called PLAN-IT, an expert system for schedule planning [Grenander, 1985]. The Deviser system figures out and plans the steps necessary to perform an action, whereas PLAN-IT decides when to schedule the actions. It is anticipated that PLAN-IT will be tested on Spacelab scheduling.

The NAVEX expert system has been developed at NASA Johnson to perform navigation functions during Space Shuttle reentry [Marsh, 1984a; Marsh 1984b]. Systems also exist for performing shuttle electrical system checks during prelaunch ground preparations [Marsh, 1984b]. Work at Ford Aerospace has been investigating the use of expert systems to automate spacecraft ground command and control functions [Wagner, 1983]. Boeing has been interested in expert systems for cruise missile automated mission planning [Jardine and Shebs, 1983] and for space

station operations [Stein, et al., 1986] including environmental, life support, and power systems [Marsh, 1984b].

The AMPASES expert system is under development by GE and TRW [Jakubowicz, 1985] for both interactive and automated space station mission planning. The MITRE Corporation is also developing an expert system called KNEECAP for crew activity planning aboard the space shuttle [Mogilensky, et al., 1983]. KNEECAP is a frame-based system derived from the architecture used by MITRE for their KNOBS expert system, which was constructed to aid in support of Air Force tactical air mission planning [Engleman, et al., 1983; Scarl, undated]. KNEECAP is intended to primarily provide an interactive scheduling capability. MITRE is also developing the EMPRESS system to assist in the planning of processing activities for cargo manifested to fly on the Space Shuttle [Hankin, et al., 1986].

A crew activity planning system does not in general entail many of the problems encountered by planning expert systems such as the Deviser, STRIPS, or MOLGEN. In such systems, it is necessary to generate, using a set of operators, each of the steps necessary to achieve a goal, and to then break the planning task into a hierarchy of subgoals. In the crew activity planning problem, however, each of the steps is given, and the task is to arrange these steps (in time) so that they do not violate any of the constraints (e.g., crew, time, and resource constraints) which are imposed upon them. These constraints are often much more complicated than those encountered in the blocks world, and the most difficult task is often to mutually satisfy these constraints.

On the other hand, several other difficulties faced by other planners are shared by a crew activity planner. If a situation is very complicated, the planner must be able to focus on the most important considerations. Additionally, in a scenario where there are more tasks to (ideally) perform than are physically possible, then the planner must be able to prioritize the tasks to be scheduled. Interactions between operations to be scheduled often occur, and the planner must be able to recognize these interactions and cope with them. Further, often the planning context will only be approximately known (e.g., some of the parameters may be random variables), so that the planner must operate in the face of uncertainty. This requires preparing for contingencies. Finally, if a plan is to be carried out by several people, coordination is required [Hayes-Roth, et al., 1983].

2.3 An Overview of Planning and Scheduling: Operation Research Techniques

The branch of operations research known as deterministic scheduling theory encompasses mathematical techniques for finding an optimal scheduling of a number of tasks among several processors (in this case, crewmembers). This field dates back to the 1950's, but has been the subject of intensive investigation over the past several years [Conway, et al., 1967; Graham, et al., 1979; Dempster, et al., 1982]. A good overview of the field is provided in the survey paper by Lawler, Lenstra, and Rinnooy Kan [Lawler, et al., 1981]. Scheduling subject to resource constraints is surveyed in [Blazewicz, et al., 1980] and algorithms are presented and compared in [Pritsker, et al., 1969; Davis and Patterson, 1975; Talbot and Patterson, 1978; and Chang and Sielken, 1980].

NASA has performed several studies involving space applications for scheduling theory. The Crew Activity Scheduling Program (CASP) was developed to aid in scheduling of Apollo missions (Murphy, et al., 1968). Fisher and Jaikumar developed an algorithm for the scheduling of Space Shuttle launches. The algorithm selected mission launch times that minimize the number of missions flown late, and satisfy early start time and resource constraints [Fisher and Jaikumar, 1978]. A number of systems have been developed at NASA for scheduling, including the Fast Automated Scheduling Technique system (FAST), the Marshall Interactive Planning System (MIPS), the Manned Activity Scheduling System (MASS), the Viking Lander Sequence of Events Scheduler (LSEQ), the Viking Lander Command Sequence Predictor (LCMSM) and the Crew Activity Planner (CAP) [Hitz, 1976].

Perhaps the most applicable research pertaining to space station crew activity planning are two studies commissioned by NASA during 1980 and 1981 [Grone and Mathis, 1980; Mathis, 1981]. Grone and Mathis describe the dispatching system used by NASA and the problem which they were addressing:

"The problem under consideration is that of scheduling a set of experiments to be performed on a given Spacelab mission. Through a preliminary analysis, which is not discussed here, a set of compatible experiments is compiled which are hopefully to be included in a specific mission. The experiments are then converted into model sheets by the principal investigator, who is responsible for the experiment design, and a NASA project engineer familiar with the format and requirements of the model sheets. The model sheet divides each experiment into a sequence of steps, each of which has certain requirements in terms of crew, equipment, and energy usage; and may in addition require that the Spacelab be in a certain position or configuration, or have certain targets available. Examples of targets are planets, communication satellites, and data receiving

facilities on the earth's surface. If the experiment is to be performed more than once the number of performances desired is included on the model sheet. A typical Spacelab mission lasts for seven days and involves six crewmembers.

"Extremely sophisticated software is available to assist in scheduling of the experiments. The current program is designated by TLP for timeline program. This program takes the model sheets in some order and "front loads" them in this order. By front load, it is meant that the model is scheduled at the earliest possible time which satisfies all the model constraints and which does not conflict with the requirements of previously scheduled models. If it is impossible to schedule a certain model at any time the program then goes to the next model in the sequence. This program is currently being upgraded to a program denoted by ESP for experiment scheduling program, which has the capability of either front loading or back loading any given model, and which is to supersede TLP sometime in 1981.

"Even with the extremely advanced and effective software available, the problem of scheduling the experiments in an optimal way currently requires an enormous amount of pre-flight man-hours. By an optimal schedule, we roughly mean one that includes the maximal number of experiments and performances, and which makes maximal use of the available resources. Since each set of models designated for inclusion in a specific mission are only roughly sorted for compatibility, it is frequently impossible to include every performance of every experiment.

"The method previously employed has been to feed TLP various random orderings of the model sheets and have the project engineers examine the characteristics of the resulting schedules. From these examinations other orderings are suggested and fed to TLP. After months of such trial and error, the best resulting schedule is then modified and edited by hand by a group of NASA personnel familiar with the characteristics of the given mission. The current schedule for Spacelab mission one has been under development for five years. It is clear that such a time expenditure is not only economically prohibitive, but is unacceptable in light of the fact that the Spacelab program eventually envisions flying several missions a year." [Grone and Mathis, 1980]

Mathis and Grone employed a ranking algorithm in the 1980 study and achieved some success. While their algorithm produces somewhat poorer results for Spacelab mission 1 than the best that had been previously generated (after several years of effort), they were able to produce schedules superior to NASA's best existing schedules for Spacelab missions 2 and 3 [Grone and Mathis, 1980]. Mathis then tested an algorithm employing zero-one programming techniques, which, while much slower than TLP, produced results only comparable to TLP, and significantly poorer than the ranking algorithm [Mathis, 1981].

It should be noted that the Soviet Union also employs a dispatching technique for the scheduling of their Salyut work schedules. The parameters relating to each of the activities are generated and each of the experiments is given a priority. Then experiments are then spread through the flight time, starting with the experiment that has the highest priority [Blagov, 1983].

2.4 Space Station Scheduling Model

A detailed inspection of a prototypical space station flight plan illustrates the complexities involved in producing a schedule. Typically, a flight plan consists of a set of jobs, or tasks, (usually on the order of hundreds) to be performed by crewmembers (typically eight or less) over a period ranging from several hours to several months. In the case of long range planning, jobs may be higher level goals, and the planning window may be on the order of weeks to years. In general, each task has a default time for completion, although in some cases different crewmembers may have different skill levels and thus different completion times. Further, in many cases only some of the crewmembers will be rated for a specific task, and the remaining crew is effectively incapable of performing that job.

Each task requires a specific number of crewmembers to perform it, ranging from zero to the total number of crewmembers available on the space station. A task might not require any crew if it is a necessary processing or waiting step in a long sequence of interlinked tasks.

In general, the scheduler must search for solutions in an environment which contains numerous types of constraints. The following is a list of typical constraints involved in space station crew activity planning:

(Those with an * are not included in the current implementation of the MFIVE Scheduler, which was developed by this study and is discussed in Section 4).

- Each flight plan has a start time, before which no job can be planned, and an end time, after which no job may be performed.
- Each task may have an earliest, latest, or required start time and a latest end time.
- Precedence relations may exist between tasks, which stipulate that one job must be completed before a second job is begun.
- Concurrence relations may exist between tasks, which stipulate that the tasks must begin at the same time.
- Time constraints may exist between tasks which designate minimum and maximum intervals between the start of the tasks.
- Preemption (job splitting) is not allowed: the processing of any task cannot be interrupted and resumed at a later time.

- Indivisible resource constraints may apply to some jobs. For example, only one spectrograph may be available, and hence only one task requiring the use of the spectrograph could be scheduled at any particular time.
- Divisible resource constraints may apply to some jobs. For example, the total amount of power consumed by all jobs occurring at any one time may be limited. (The MFIVE Scheduler, as presented in Chapter 4, considers four types of divisible resources (the same ones used in Grone and Mathis, 1980), but in principle it could incorporate many more, without any fundamental changes to the scheduling problem. These four resources are power usage, high rate multiplex, memory usage, and data transmission.)
- Nonlinear resource constraints may apply to some jobs. For example, the MFIVE Scheduler (Chapter 4) incorporates an audio level resource which indicates whether a job is sensitive to or producing audio or vibration noise. Some task which is vibration sensitive can only be scheduled while there are no other tasks scheduled which are vibration or noise generating.
- A task may have multiple target time windows during which it can occur, and these windows may be given by the solution (either exact, numerical, or logical) of arbitrary functions of time. For example, it may be required that space station orbital parameters be within certain ranges in order to perform a specific task. The applicable equations can then be solved (either exactly, through some iterative search procedure, or through table lookup) to give time windows for the job.
- * Multiple performances of a task may be specified, with maximum and minimum time intervals between the completion time of one performance and the start time of the next performance. For example, sleep periods could be scheduled on a daily basis with an interval of 15 to 17 hours between the end of one sleep period and the start of the next.
- * Tasks may be grouped together into larger units, with maximum and minimum time intervals between various tasks and a maximum duration on the performance of the entire unit.
- * Tasks may be beneficially grouped by class (several EVA tasks could best be completed during a single EVA, for example).

The goal of the scheduler is then to produce a schedule which maximizes the weights (assigned by some user provided prioritization scheme) of the jobs scheduled (there may be more tasks to

perform than can be fit into the timeline), equalizes individual workload, minimizes total workload, and meets subjective approval of the crewmembers. A schedule which is optimal by one of these criteria may not be optimal by the other criteria, and hence a schedule should employ an algorithm which produces a "reasonable" combination of all the above criteria.

Several strategies are available for attempting to produce an optimum schedule. In theory, the problem can be cast as an integer programming problem, and solved by any of various techniques available. However, this problem is "NP-hard": it grows exponentially in difficulty with the size of the problem. Even for problems of modest size, it is much too difficult to be solved optimally. In practice, various heuristic strategies can be employed to produce schedules which are "good". As stated above, the optimization criteria are somewhat inexact (as is, to some extent, the data base on which the scheduler relies); hence getting within, say, five percent of the optimum would be considered a success.

Heuristic algorithms for solving this type of problem can be grouped into two basic classes: those which schedule each job (perhaps one at a time) without looking at the "big picture" (greedy algorithms); and those which attempt to produce a schedule for all the jobs simultaneously (global algorithms). The MFIVE Scheduler of Section 4 has been designed by this study to facilitate the testing of these algorithms to determine their speed and efficiency.

Section 3: Stowage Logistics

3.1 The Stowage Logistics Clerk

The National Aeronautics and Space Administration put the first Skylab crew in orbit in November 1973. The mission produced an extensive data base on long-term human productivity in a microgravity environment. The Skylab flight crew communicated regularly with the ground support staff at Lyndon B. Johnson Space Center in Houston, Texas. JSC was responsible for planning the crew's activities for each day of the mission, and for providing whatever assistance they could in the performance of the scheduled experiments. In their communications with JSC, the flight crew described some of the difficulties attendant to performing tasks in orbit. Much of the interaction between Skylab and JSC concerned logistical problems the crewmembers were experiencing as they attempted to follow their schedule.

Locating the equipment necessary to execute a task required a disproportionate amount of the Skylab crew's time. Objects were not easy to keep track of in the microgravity environment; if unrestrained, they floated around randomly, their paths subject only to the air currents of the life-support system. It was therefore essential to keep all items stowed, except during times of actual use. The number of items carried in the long-term Skylab flights had increased dramatically over previous shorter mission; this increase required a parallel growth in the number and types of onboard stowage locations. When objects did get stowed away, Skylab experience indicated that they were often put back in compartments other than their nominal locations, leading to a lot of frustration and wasted time in orbit and on the ground:

"The floor of the upper deck [of Skylab] was a confusing jumble of experiments because the astronauts didn't always put things away properly. Bean, the commander of the second mission, had explained over television to the third crew, while they were still on the ground, 'We just leave stuff around. We'll stow it before we go. Otherwise, you get in the business of stowing and unstowing if you're not careful' -- an unusual philosophy for a normally shipshape Navy captain. As it happened, a lot of stuff never *had* gotten put away properly. By the time the third crew had arrived, stowage was completely out of hand... The mess, which had piled up during the first two missions, slowed the third crew down as much as anything else." [Cooper, 1976]

The system NASA devised for keeping track of equipment was mechanized by a group at JSC who listened to the air-to-ground communications and recorded the locations of each piece of equipment mentioned. This type of real-time monitoring procedure was not equal to the task:

"There were some forty thousand items stashed away in over a hundred cabinets in the space station, and Pogue bitched that none of them was ever stowed where a person might logically expect to find them. Although there were six men and a computer in Houston whose sole purpose was to help the astronauts keep track of items in the space station, the system, which had been breaking down since the beginning of the second mission because of the progressive failure of the crews to report where they put things, had now collapsed altogether." [Cooper, 1976]

There was another disadvantage to NASA's answer to the stowage problem: it required the crewmembers to consult the ground-based task force before almost every activity. There was evidence that the performance of the flight crew was adversely affected because they were too dependent upon the ground support team for their everyday operations. A 1983 NASA Summer Workshop published a report concluding that a future space station crew's effectiveness would be increased by giving them more control over their own destiny in day-to-day matters:

"What is the appropriate degree of control of ground personnel over station crewmembers? In recent years, the trend in American management has been towards greater decentralization -- more autonomy -- among organizational subunits. Effectiveness gains appear through (1) closer attention to problem-solving and decision making information, (2) higher motivation, (3) significant improvement in quality, (4) greater adaptability, (5) less surveillance (lighter supervision, smaller staffs, less overhead), and (6) faster and more creative decisions." [Johnson, et al., 1985]

The NASA Space Station Program presents a much greater logistical challenge than Skylab because of its greater magnitude and complexity. Figure 1 shows a possible layout of the modules that comprise the living sections of the space station. Although the design is not yet final, it has been proposed that the initial configuration of the manned portion of the station will consist of five cylindrical modules, grouped to permit multiple access paths between modules. Two Habitation Modules (Figures 2 and 3) will contain all facilities necessary for crew sustenance and station operation (the "core" activities). "Mission" activities, such as performing experiments and materials processing and manufacturing, will take place in two Laboratory Modules (Figures 4 and 5). A Logistics Module will serve as the station's "closet": it will store resupply items, spares, etc. Berthing ports, included either on the modules or on separate connector structures, will be designed to permit new modules to be attached to the configuration at some later date. While it is

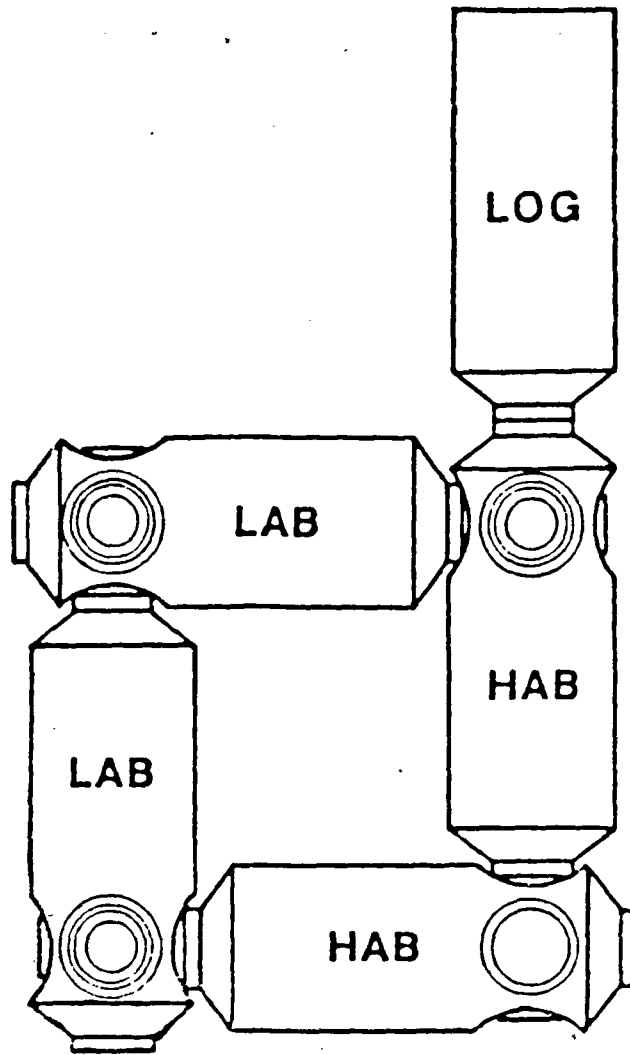


Figure 1 : SPACE STATION MODULE CONFIGURATION

ORIGINAL PAGE IS
OF POOR QUALITY

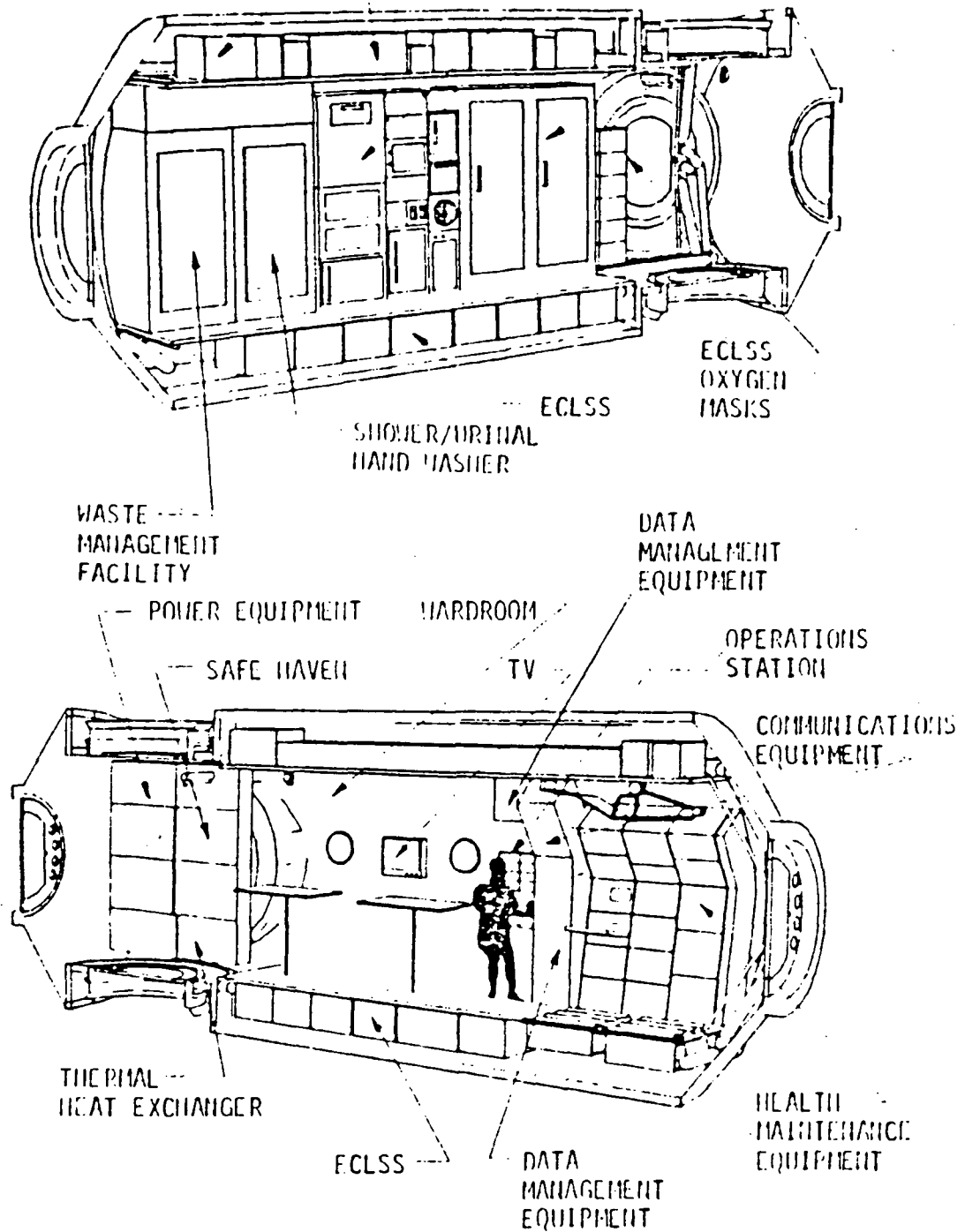


Figure 2 : HAB 1 MODULE

- Reprinted from Space Station Reference Configuration Description, Systems Engineering and Integration, Space Station Program Office, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1984.

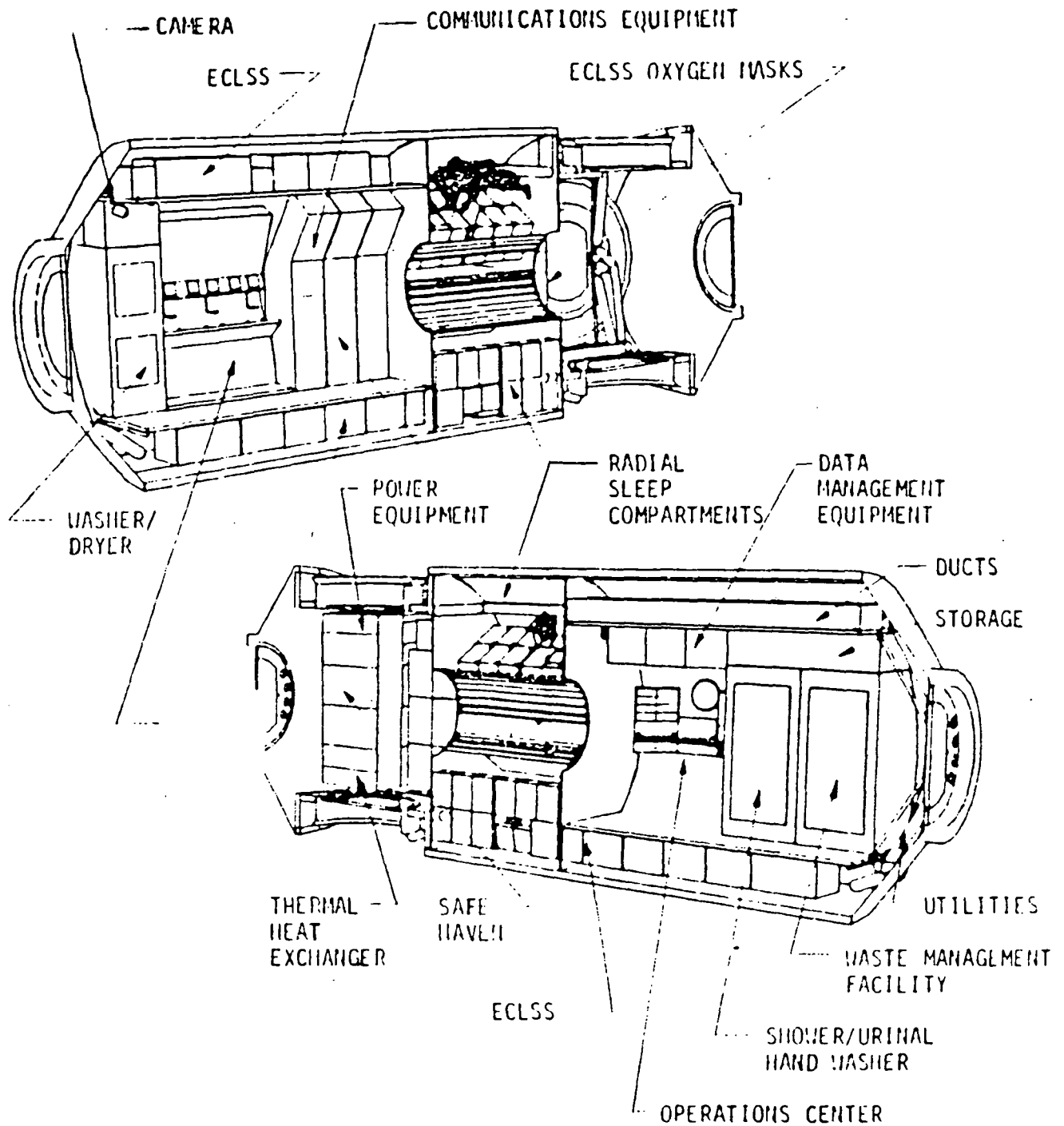


Figure 3 : HAB 2 MODULE

● Reprinted from Space Station Reference Configuration Description, Systems Engineering and Integration, Space Station Program Office, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1984.

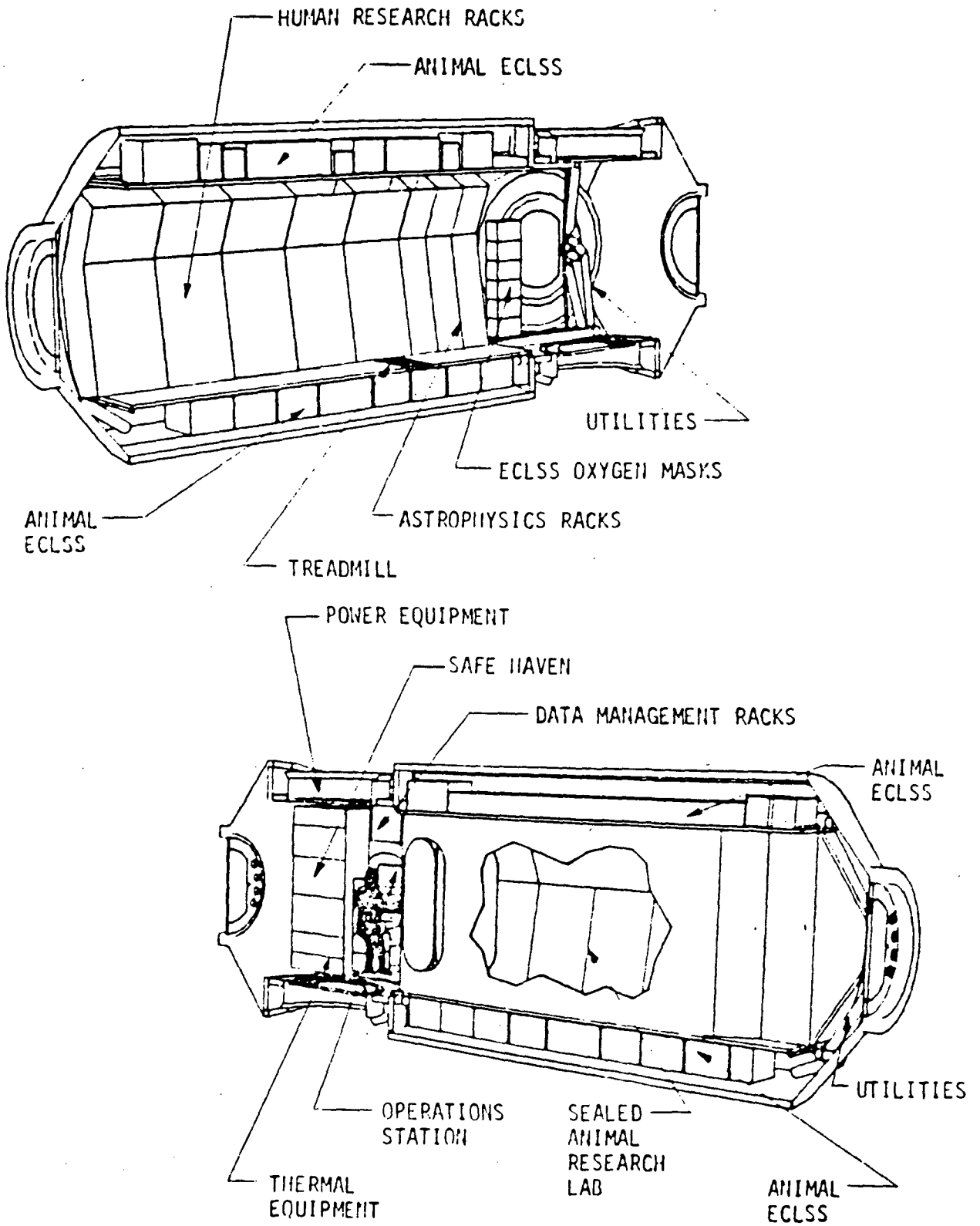


Figure 4 : LIFE SCIENCES LAB

• Reprinted from Space Station Reference Configuration Description,
 Systems Engineering and Integration, Space Station Program Office,
 NASA, Lyndon B. Johnson Space Center, Houston, TX, 1984.

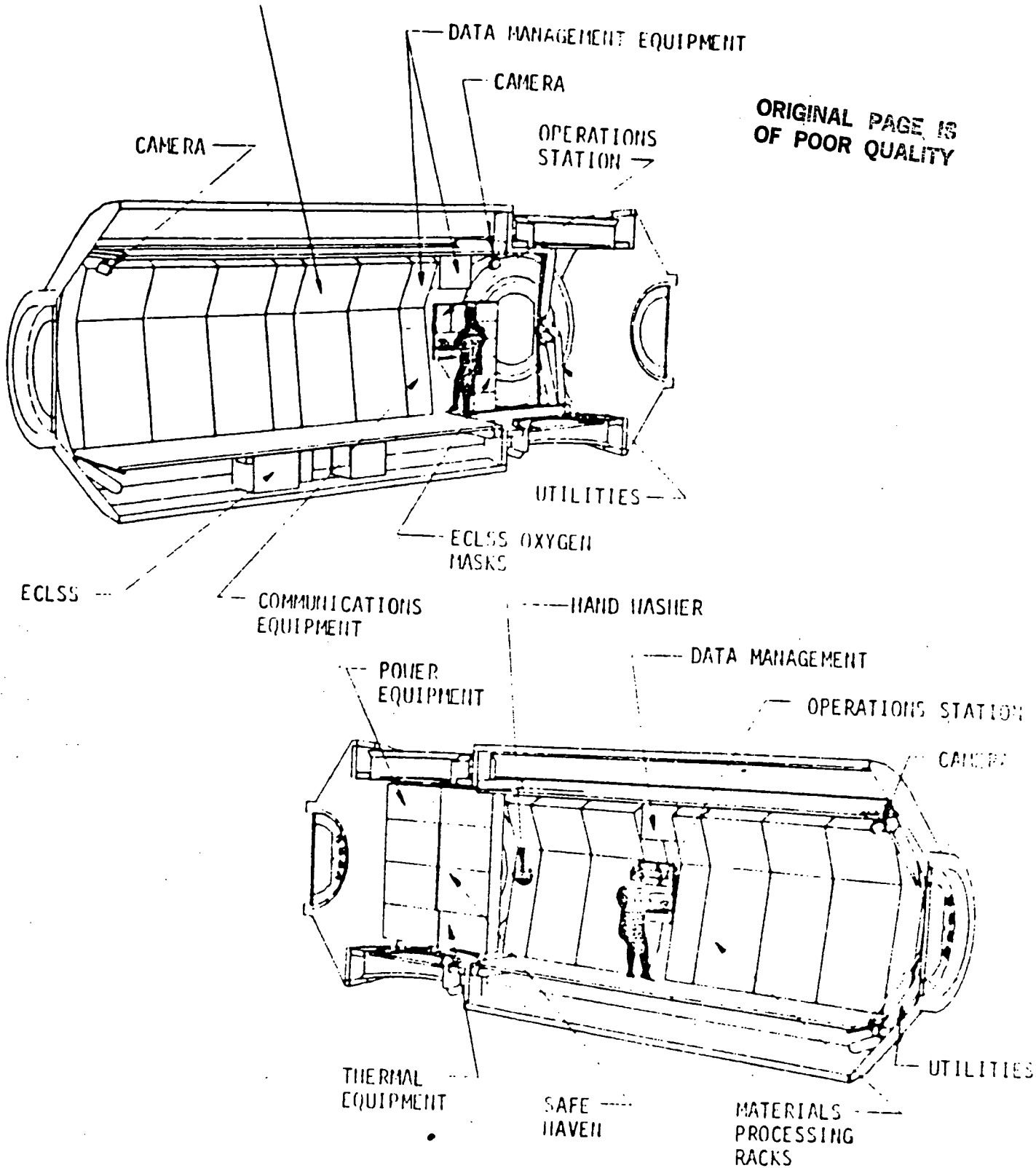


Figure 5 : MATERIALS PROCESSING LAB

- Reprinted from Space Station Reference Configuration Description, Systems Engineering and Integration. Space Station Program Office, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1984.

recognized that current space station configurations have changed from the details presented here, the significance to be drawn from this presentation is that the space station will have a large volume, intricately laid out, with continual dynamic changes throughout the life of the station.

The Laboratory Modules will contain dedicated racks of different sizes for storing the equipment used in the various activities; the Habitation Modules will house storage bins for food, exercise equipment, etc. Most of the rest of the stowage bins will be of a standard volume (probably two cubic feet); these will be liberally distributed throughout each module (Table 1). The Logistics Module will consist almost wholly of bins of this size [Space Station, 1984].

3.2 Stowage Management System Specifications

It has been shown that stowage management problems exist in an orbiting spacecraft, and that this situation has in the past led to a decrease in the flight crew's productivity. Previous attempts at using ground-based personnel to solve these problems have not been satisfactory. A potential solution to the problem of stowage management is the creation of an interactive onboard computer system that would keep track of stowage allocation and aid the flight crew in locating missing equipment. This system must obey the following requirements:

1. It should provide the location of the missing piece of equipment; failing that, it should provide a list of reasonable guesses. One of these possible locations will be the item's nominal stowage container.
2. It should give the name of the item's last user and the module in which it was last used. This is facilitated by integrating the stowage clerk with the crew task scheduler.
3. It should help keep track of the functional status of all equipment: operational, broken, broken and under repair, disabled (e.g., an empty fire extinguisher is disabled), location unknown, and lost (i.e., nonrecoverable).
4. It must be user-friendly. This system is proposed as an aid to the crewmembers, not as an additional task. Some user interaction will be necessary, but it should be kept to a minimum.

3.3 Expert System Concepts

There are several possible methodologies that could be used to provide absolute knowledge of the location of a piece of equipment. One option is to simply provide a large database, to which the

crewmembers would report when they locate an object. This idea suffers from the same problems as NASA's original solution: it is too cumbersome to be of much use, and can easily fall out-of-date, due to disuse. (However, the advent of reliable voice-recognition systems could make this option more attractive.) Another possibility would be to place bar codes on each object (similar to grocery items) and install laser scanners on each bin that would feed their information into a central computer. This would probably be prohibitively expensive, although a company called Infocel (Rockville, MD) has recently developed a reasonably inexpensive bar code system for tracking paperwork as it moves about a company [Glazer, 1985]. Alternatively, laser scanners could be installed only at the major transportation nodes, to monitor the movement of equipment. Although this would not provide absolute knowledge of an item's location, it would provide valuable input to a stowage management system. However, the viability and utility of such a system would require careful study before it could be integrated into a stowage logistics clerk.

It appears that it is unrealistic and infeasible to provide a system that will have absolute knowledge of each item's location. It follows that any logistics clerk will have to employ some "smart" reasoning algorithm to provide good guesses as to an item's location. A program with this level of inferential capability is generally referred to as an expert system. An expert system is defined as one that:

- handles real-world, complex problems requiring an expert's interpretation
- solves these problems using a computer model of expert human reasoning, reaching the same conclusions that the human expert would reach if faced with a comparable problem. [Weiss and Kulikowski, 1984]

An expert's problem-solving ability may be represented as being composed of two parts: experience and reasoning power. Experience refers to the knowledge an expert has gleaned after extensive involvement in the area of speciality. Experience produces knowledge of what data is required to diagnose a problem, and therefore what tests should be made. Reasoning power refers to the expert's ability to recognize relationships between the evidence that has been gathered and the database of knowledge gained from experience. Based on these relationships, the expert can draw conclusions that lead to a solution. The power to make these inferences can be expressed as a set of rules; heuristics rather than well-defined algorithms lie at the base of human expertise. An attempt will be made to simulate this process of inference in the stowage logistics clerk.

3.4 Expert System Design

The primary task in the design of an expert system is to define the problem that is to be solved, and then to model the reasoning strategies by which a human expert would approach that problem. Weiss and Kulikowski, in considering expert systems that deal with the category of *classification* problems, define some simple design techniques [Weiss and Kulikowski, 1984]. Classification problems include diagnosis, interpretation, or giving advice. The stowage logistics clerk is an advisory program.

The classification problem model used in expert systems consists of three parts: a list of observations, a list of conclusions, and a set of rules which relate the observations to the conclusions. A *production system* is one which attempts to solve classification problems by applying *production rules*. In general, there are three components to a production system: (1) a set of production rules; (2) a global data base; and (3) a control system.

The production rules make up the expert's knowledge base. They are organized as a set of heuristics that represent all that is known about solving the classification problem. The global database is the store of relevant information that has been gathered. In an expert system that performs medical diagnosis, for example, the global database is made up of observations concerning the patient's condition. The control system, or *inference engine*, controls the application of the production rules to the global database. There are two methods by which the control system can step through its reasoning: forward chaining and backward chaining.

Forward chaining is also known as data-directed reasoning. In this case, the system chooses which production rules it will apply based on the evidence at hand. The conclusions that it reaches and/or the subsequent data that it receives determine the next production rule to be invoked. A backward chaining, or goal-directed, system begins with a set of possible answers, and evaluates them based upon all of its production rules. The appropriate answer or answers are those that survive the control system's interrogation. In the medical example, the goals could be possible diagnoses that are eliminated one by one until the final answer is decided upon.

In the stowage logistics problem, the system must consider a list of possible places in which a tool might be found, and evaluate each bin's "appropriateness" as a candidate location. This problem differs from many other expert systems cases in that there are no specific experts whose

advice can be modeled; few people are "experts" at finding missing equipment aboard space stations. Common sense must therefore dictate the algorithms that will be used when deciding where to look; the heuristics that are chosen will form the system's set of production rules. The global database will consist of all information required by the control system concerning the item being searched for and the stowage bins in which it might be located. The control system will employ something similar to a backward chaining strategy, because each bin in the space station is a possible answer. The bins will be evaluated according to probability by application of the production rules.

3.5 The Design of the Stowage Logistics Clerk

In a way, everyone can consider themselves to be an expert on the subject of finding personal items that are missing: everyone has plenty of experience at it. However, few people have ever considered the logical thought processes that are used to locate missing items. Logical places to look begin with the last place the item was seen or used, and the place it is normally stored. If the item is not found in any of these primary locations, a global search ensues of every place in which the item has been seen, leading ultimately to a search of all possible locations. However, complications arise when this experience is extrapolated to a weightless environment. Objects in microgravity do not stay where they are placed unless they are restrained. On the space station, most items to be located will be stowed in a (presumably) opaque bin. Clearly, such global searches will be extremely costly in terms of crew time; careful thought must be given to minimizing the scope of these searches. Given this definition of the problem, the following search rules may be useful [Kranzler, 1986]:

1. **No object may be located in a container in which it cannot fit.**
2. **Objects are likely to be left in a container that is the same size as the container in which they are normally found.** In microgravity, items stowed in a disparately large volume tend to float freely within, and to float out of the container upon opening (this tendency has been termed the "jack-in-the-box effect".) It is presumed that crewmembers would attempt to minimize this problem by stowing objects in locations of comparable dimensions.
3. **An object is likely to be left in or near one of two locations: where it belongs, and where it was last used.** An astronaut looking for a camera might expect to find it near its nominal stowage bin, or near the site at which it was most recently used.

4. **Items in common use will not be placed in long-term storage.** Astronauts will only look in the Logistics Module, where spares and resupply items are kept, when the nominal copy of an item is broken or can't be found; in that case, it should be sufficient to provide the user with the spare's default location. Other than that, the remote location of the logistics module makes it a rarely-traveled area and a bad place to look for missing items.
5. **Objects with similar purposes are usually stored together.** The space station flight crew will probably be under a great deal of time pressure to perform activities according to schedule. It is difficult to envision an astronaut who has just finished a task scurrying about to put everything away in its proper place, if this involves a number of widely scattered bins. He will probably stow all of the related equipment in the same bin, or in adjacent bins.
6. **If an item has been found in a particular location many times, then it is highly likely that it will be found there again.**

These heuristics will require a database on which to operate. The database will need to maintain information on the following:

- bin size
- bin location
- nominal bin contents
- space station configuration (i.e., the connectivity and physical layout of station modules)
- module type (i.e., habitation, laboratory, or logistics)
- tool size
- quantity (number of copies) in stock
- nominal locations of each copy
- operational status of each copy
- jobs for which the tool is required
- the crewmember who last used the tool
- the place in which the tool was last used
- the bins in which the tool has previously been found

The above parameters may be specified by a number of sources. Space station configuration and bin sizes and locations will probably be provided by ground operations. The crewmembers must be able to access all relevant information and should (at their option) be able to tell the system the bins in which missing tools have been found. An advantage of having a scheduler and a logistics clerk share the same database is that information concerning the place of last use, and the crewmember who last used the tool, can come to the logistics clerk directly from the crew activity

scheduler. Additionally, crewmembers can be given information concerning the locations of needed tools along with their schedules and job instructions. The Skylab stowage system described earlier was a paper-bound equipment manifest that was nearly useless because it was never up-to-date. If provided with a truly user-friendly management system, the crew can be reasonably expected to update the database when new tools are added and missing tools are found.

The above heuristics and database form the basis of the production system used by the MFIVE Stowage Logistics Clerk (Section 4.2.3). More accurate inference could be possible by asking for more information from the user, but this could bog down the operation of the system and violate the user-friendly specification. This illustrates an inherent tradeoff between accuracy and tractability.

Section 4: The MFIVE Crew Scheduler/Logistics Clerk

4.1 Developmental Goals

The primary goal of this effort has been to design a computer scheduling system/logistics clerk which can model the space station environment to allow the testing and optimization of algorithms for activity scheduling. An emphasis has been placed on demonstrating a system that is easy to operate and inexpensive. MFIVE development started on a mainframe, was transported to an IBM Personal Computer, and was then finally implemented on an Apple Macintosh. This transfer of the system from computer to computer generated some significant delays in the study (about 3/4 year) but the end result clearly indicates that the delay was worthwhile. For example, the initial mainframe system had complex natural language interface for querying the system for data about the crew and space station. On the Macintosh, this interface is completely unnecessary; it is much easier for the user to click icons and scroll windows with a mouse. The superior graphics and user interfaces available on the Macintosh have proven invaluable. Further, the entire hardware necessary to run the system can now be purchased for under \$2000.

Some of the major limitations of the Macintosh (as compared to something like a LISP machine or an Apollo computer) include its (relative) slowness and its small screen size, which limits the size of the work area and the amount of data which can easily be presented to the user. From the start, it was intended to complement, rather than compete with, other projects, such as KNEECAP and AMPASES. These programs are implemented on much more sophisticated (and costly) LISP machines.

There are several other software efforts which have designed scheduling software for personal computers [Freedman, 1985], but these projects usually are limited to standard PERT/CPM techniques which cannot accommodate the complexities present in space station scheduling, such as resource constraints. Available scheduling systems include, among others, the *Total Project Manager* from Harvard Software; *Pac III* from AGS Management Systems; *PMS-II* from North America Mica; *MicroGANTT* from Earth Data; and *MacProject* from Apple Computer.

MFIVE makes many simplifications of the space station environment, but these instances do not fundamentally alter the nature of the scheduling environment. For example, in the description

of a space station experiment, MFIVE has a parameter describing the amount of power (in Watts) used by the experiment. In a more realistic scenario, one would have to specify many parameters, such as nominal power, peak power, operation periods, power type (AC/DC) and voltage, etc. The inclusion of this information would not fundamentally change the nature of the scheduling problem: one would just have to perform more complicated checks to determine when it would be feasible to schedule the experiment. There is no reason why this type of information could not be added to the MFIVE system. However, including this type of data now would only slow program operation and consume substantial programming time, while not really adding anything "new" to the system; this is not in keeping with the goals of this study. Further, other studies, such as KNEECAP and AMPASES, are addressing these aspects of the scheduling problem in much greater depth.

It was originally anticipated that a LISP/PROLOG-like approach would be desirable for generating English language rules on which to base the scheduler. Further investigation, however, proved otherwise. A LISP-like approach is very well suited to providing some user interfaces (such as explanations of reasoning chains), and for doing things like constraint satisfaction via rule checking and inference. The process of finding an optimal schedule, however, involves highly mathematical algorithms, which are usually not well suited to English-based rules. There is, of course, no reason that these algorithms cannot be implemented in LISP; it is just that it is unnecessary (and often more complicated) to do so.

This study therefore chose APL as the implementation language for MFIVE. The high level features of APL allow very quick program testing and development. APL is a highly mathematical computer language which allows easy vector/matrix manipulation, and is well suited to implementing operations research algorithms. Being interpreted, however, APL is rather slow compared to some other compiled computer languages. It is not anticipated that APL would be the language of choice for an operational system; it is, however, well suited to the goal of algorithm development and testing. The APL interpreter used is called PortaAPL, and is developed for the Macintosh by Portable Software, 60 Aberdeen Avenue, Cambridge, MA 02138.

4.2 MFIVE Operation

MFIVE is, at the highest level, organized into three modules (Figure 6): the PRIME module serves as a data base management system, allowing the user to enter and modify all data pertinent to

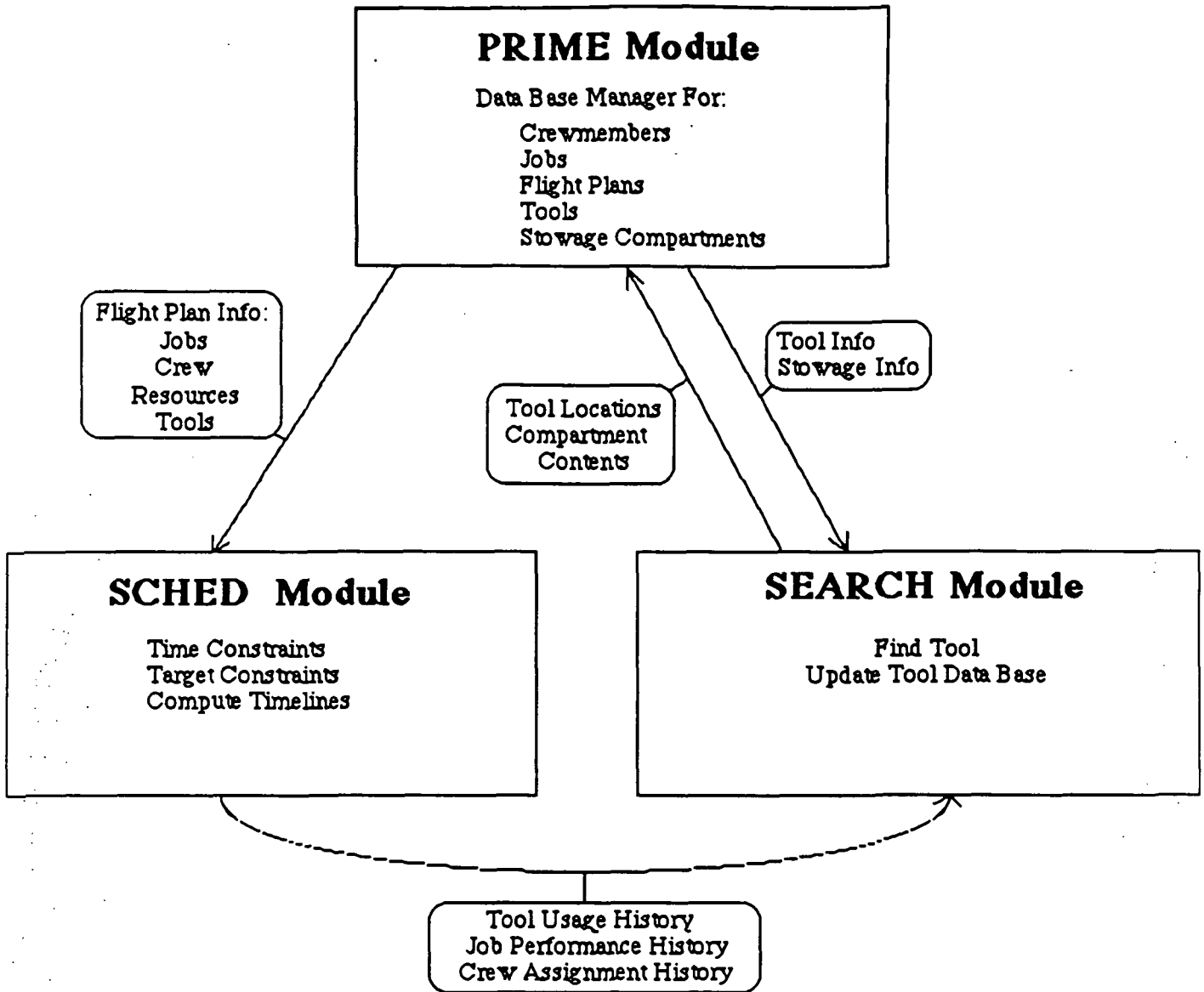


Figure 6: MFIVE Organizational Structure

the crewmembers, the jobs they perform, the tools used, the stowage layout of the space station, and the assignment of crew and jobs into organized flight plans; the SCHED module allows the interactive and automatic scheduling of a flight plan; and the SEARCH module performs searches for tools which are difficult to locate. The following sections illustrate operation of the MFIVE program.

4.2.1 The Data Base Management System

MFIVE operation is initiated by clicking the mouse on the PRIME icon in the Macintosh finder window. The user is then presented with six icons (Figure 7). Selecting the EXIT icon allows the user to terminate MFIVE operation and return to the APL environment. Selecting any of the other icons allows the user to access the data base which the icon represents. For example, selecting the icon labelled CREW will call up a generic Crew Information Form and prompt the user to enter the name of a crewmember (Figure 8). At this point, the user has four choices: to type in the name (or nickname or ID number) of a crewmember which MFIVE already knows about; to type "help," in which case MFIVE will present a window allowing the user to select a crewmember from a list (Figure 9); to type "new," in which case MFIVE will prompt the user to enter the name of a new crewmember; or to hit the return key without typing anything, in which case MFIVE will return the user to the opening program configuration (Figure 7). Once MFIVE has the name of a crewmember, that crewmember's information form will be called to the screen, displaying and allowing revision of all data pertinent to that crewmember (Figure 10).

As Figure 10 shows, some of the data applicable to a crewmember is displayed directly on the screen (e.g. date of birth), in which case the information can be modified by clicking on the current entry, and then entering a new value in response to a prompt by MFIVE. Other information (e.g. crewmember background information) is initially hidden, and is brought to the screen for revision by clicking on the applicable box (Figure 11). The next sections outline the details of the various information forms.

The Crewmember Information Form (Figure 10)

The NAME slot: Clicking on the name of a crewmember allows the user to revise the crewmember's name or to delete the crewmember from the system (by typing "delete").

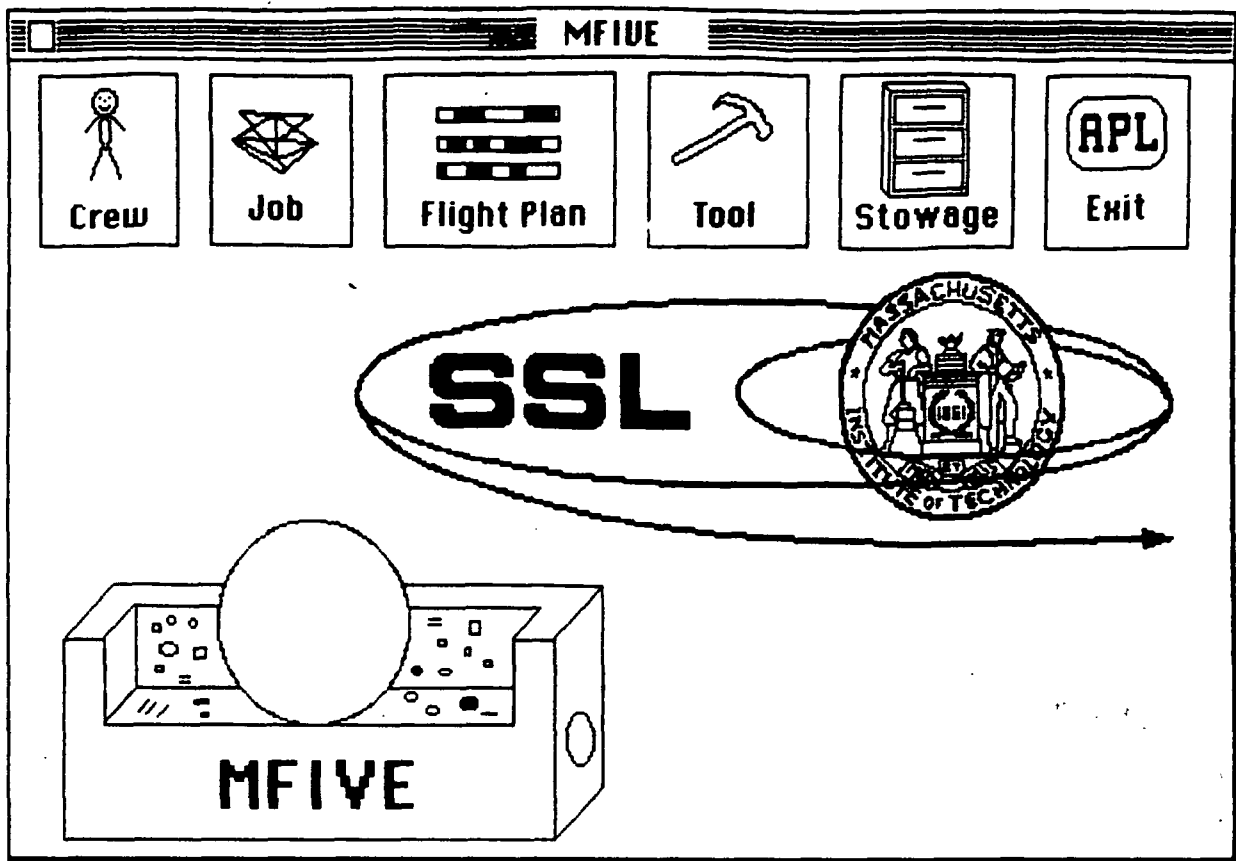


Figure 7

The image shows the same 'MFIVE' software interface as Figure 7, but with the 'Crew' menu item selected. The main content area is titled 'Crewmember Information Form'. It contains the following text labels for data entry:

- Name:
- Number:
- Date of Birth:
- Mass (kg):
- Height (cm):
- Assignment Start Date:
- Assignment End Date:
- Rank:

Below these labels are three rectangular buttons stacked vertically:

- Performance Data
- Background Information
- Nicknames

Figure 8

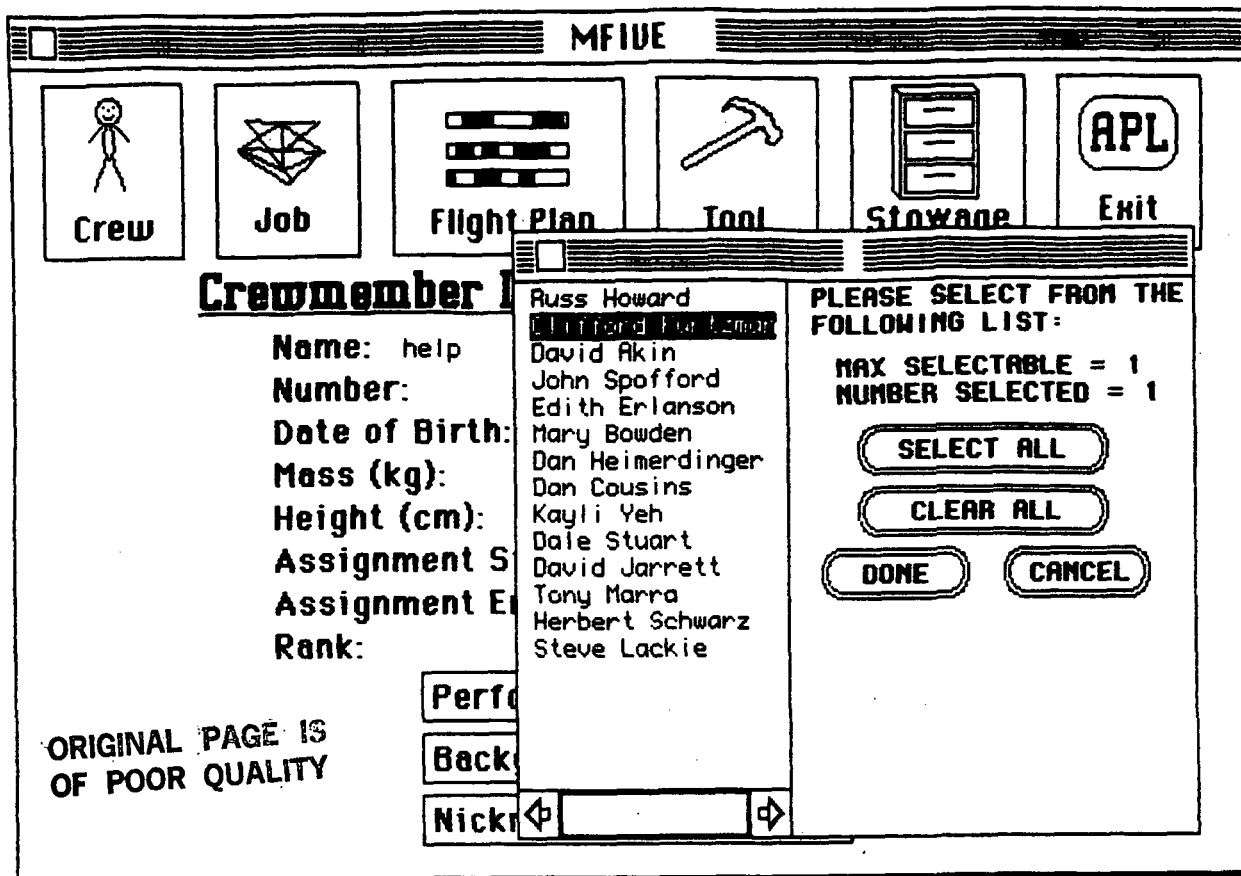


Figure 9

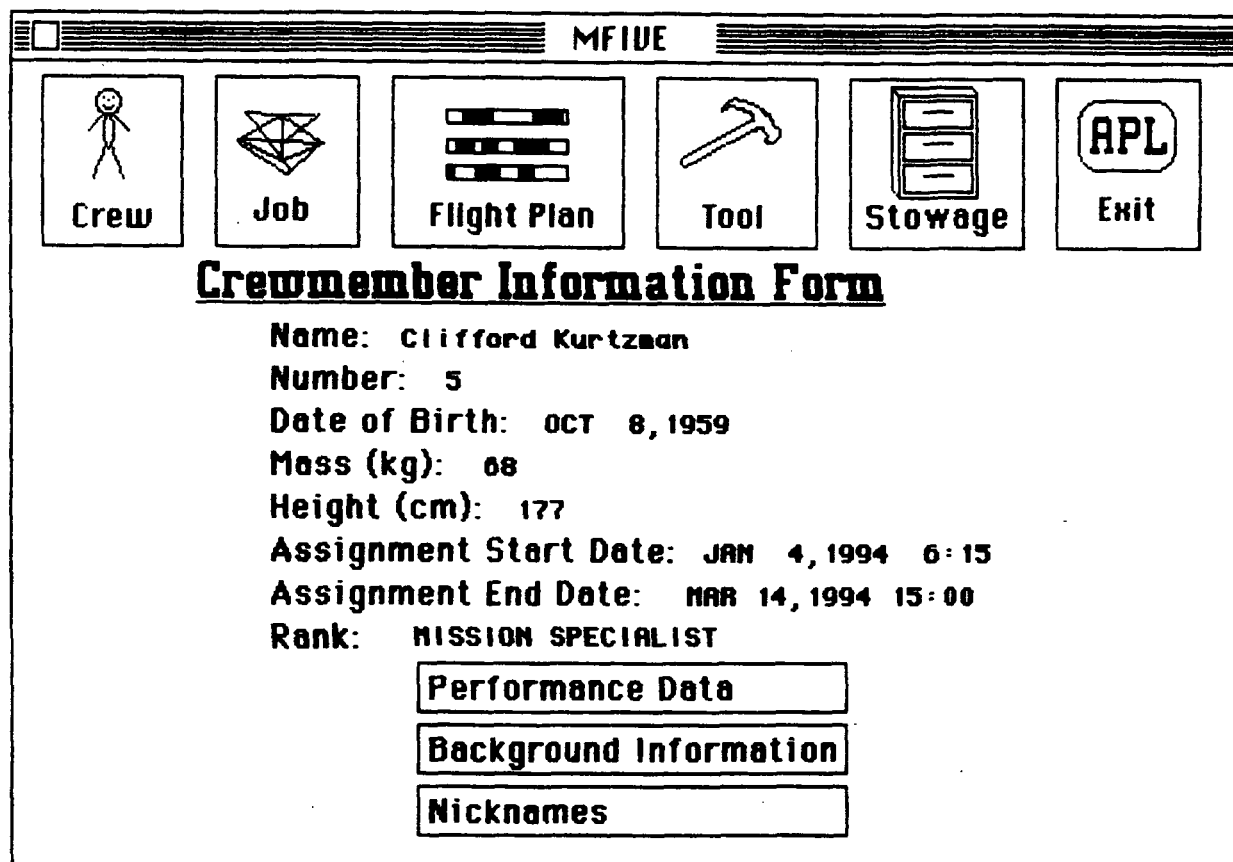


Figure 10

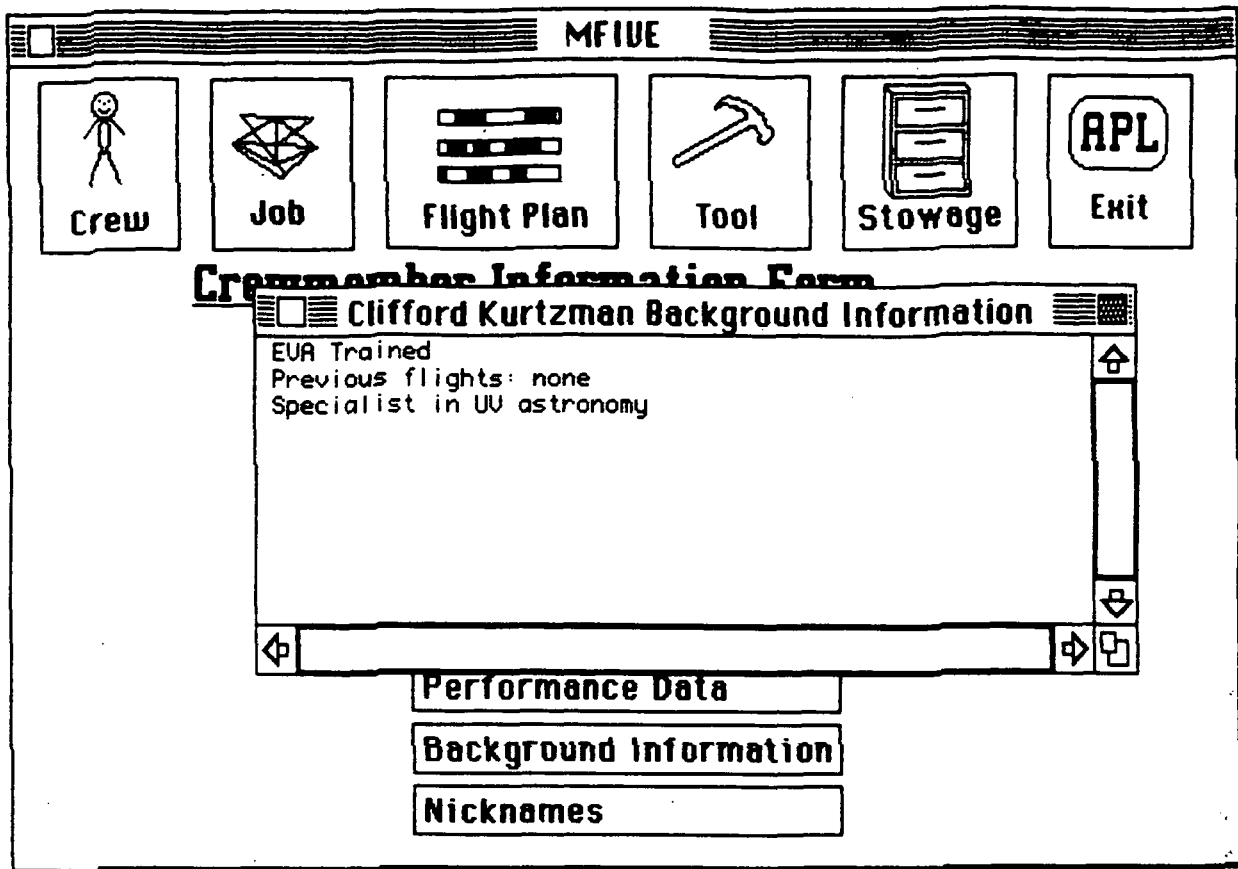


Figure 11

The **NUMBER** slot: This number is an ID number for the crewmember assigned by MFIVE. It is not modifiable by the user.

The **DATE OF BIRTH** slot: This information is revisable by clicking on the date, or can be deleted by typing "delete" (the delete option is typical of most information slots, and will not be explicitly mentioned further).

The **MASS** slot: This specifies crewmember mass in kilograms.

The **HEIGHT** slot: This specifies crewmember height in centimeters.

The **ASSIGNMENT START DATE** and **ASSIGNMENT END DATE** slots: These specify the first and last dates at which the crewmember becomes assigned to the space station.

The **RANK** slot: This specifies the status of a crewmember, e.g. Mission Specialist, Commander, etc. A listing of available options, from which the user can then make a selection, is given by typing "help".

The **PERFORMANCE DATA** box: Clicking this box displays a scrollable window showing the time it takes the crewmember to perform each of the jobs (Figure 12). In general, each job has a default performance time, which all crewmembers are initially given. Clicking on a particular job allows the user to change or delete the performance times. An asterick next to a performance time indicates that the time listed is the default time. A crewmember for which the performance time has been deleted cannot be assigned that task by the scheduler. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The **BACKGROUND INFORMATION** box: Clicking this box displays a scrollable and resizable window containing information about the crewmember (Figure 11). Clicking within the box moves the cursor to allow editing and addition of text. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

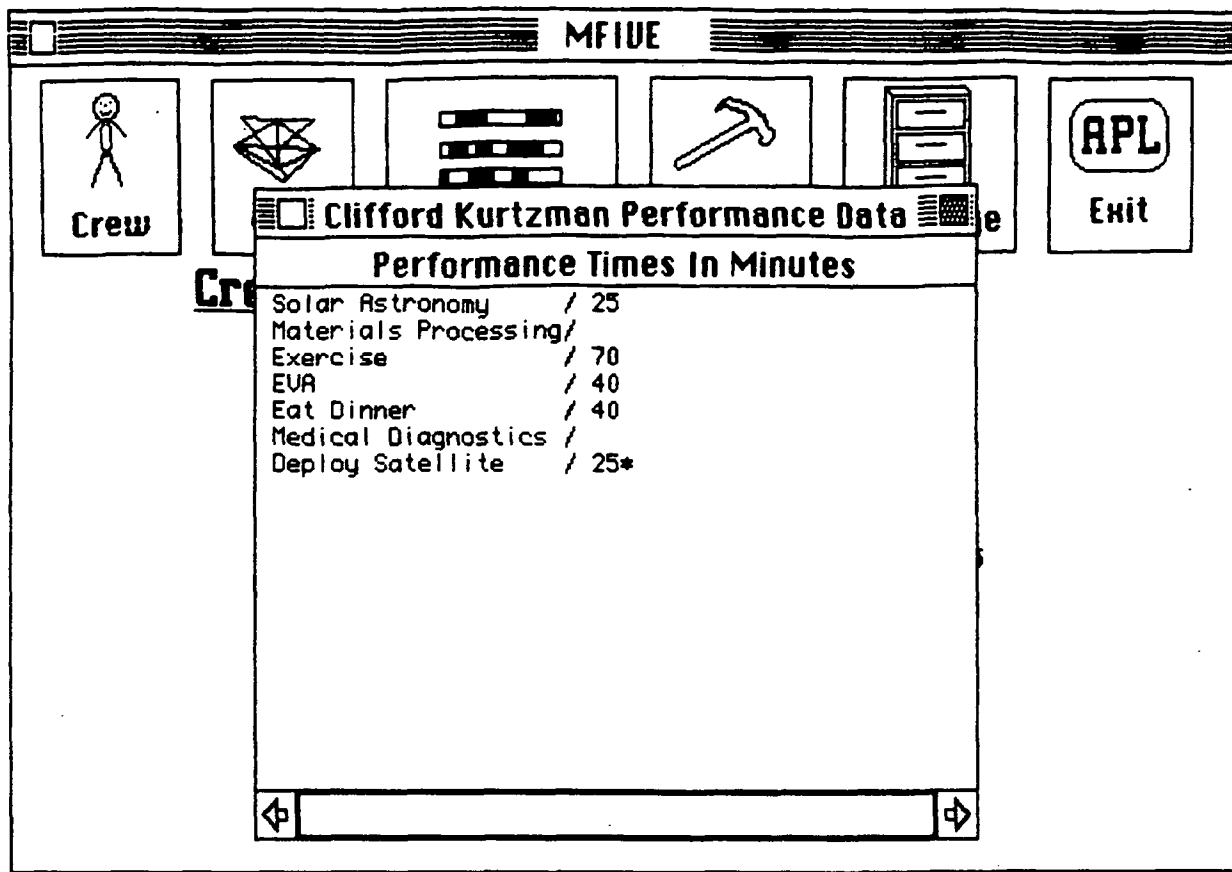


Figure 12

The NICKNAMES box: Clicking this box displays a scrollable window which contains the nicknames of a crewmember (Figure 13) that are recognized by MFIVE. By clicking on the REVISE button and then on a nickname, one can make revisions. Deletions are made by clicking on the DELETE button and then on the nickname to be deleted. The other buttons are self explanatory. The white box at the upper left corner of the window is the same as the DONE button.

The Job Information Form (Figure 14)

The NAME slot: Same as for the Crewmember Information Form.

The NUMBER slot: Same as for the Crewmember Information Form.

The DEFAULT JOB TIME slot: This slot specifies the baseline time to complete the job. This information can be superceded by specific information for any particular crewmember (see PERFORMANCE DATA, below).

The CREWMEMBERS REQUIRED slot: This slot specifies how many crewmembers are necessary to perform the job.

The POWER USAGE, HIGH RATE MULTIPLEX, COMPUTER MEMORY, AND DATA TRANSMISSION REQUIREMENT slots: These slots specify how much of each of these resources is used by the job. Resource usage is taken to be constant throughout the entire execution of the job.

The AUDIO LEVEL slot: This slot indicates whether the job is sensitive to or producing audio or vibration noise. Possible values are: SENSITIVE for jobs which cannot be performed during excessive noise or vibration; NEUTRAL for jobs which do not produce and are unaffected by noise or vibration; GENERATING for jobs which produce noise or vibration, and UNKNOWN. Typing "help" calls up a window of these options for the user to make a selection.

The PERFORMANCE DATA box: Clicking this box displays a scrollable window showing the time it takes each of the crewmembers to perform this job (Figure 15). (This is the same

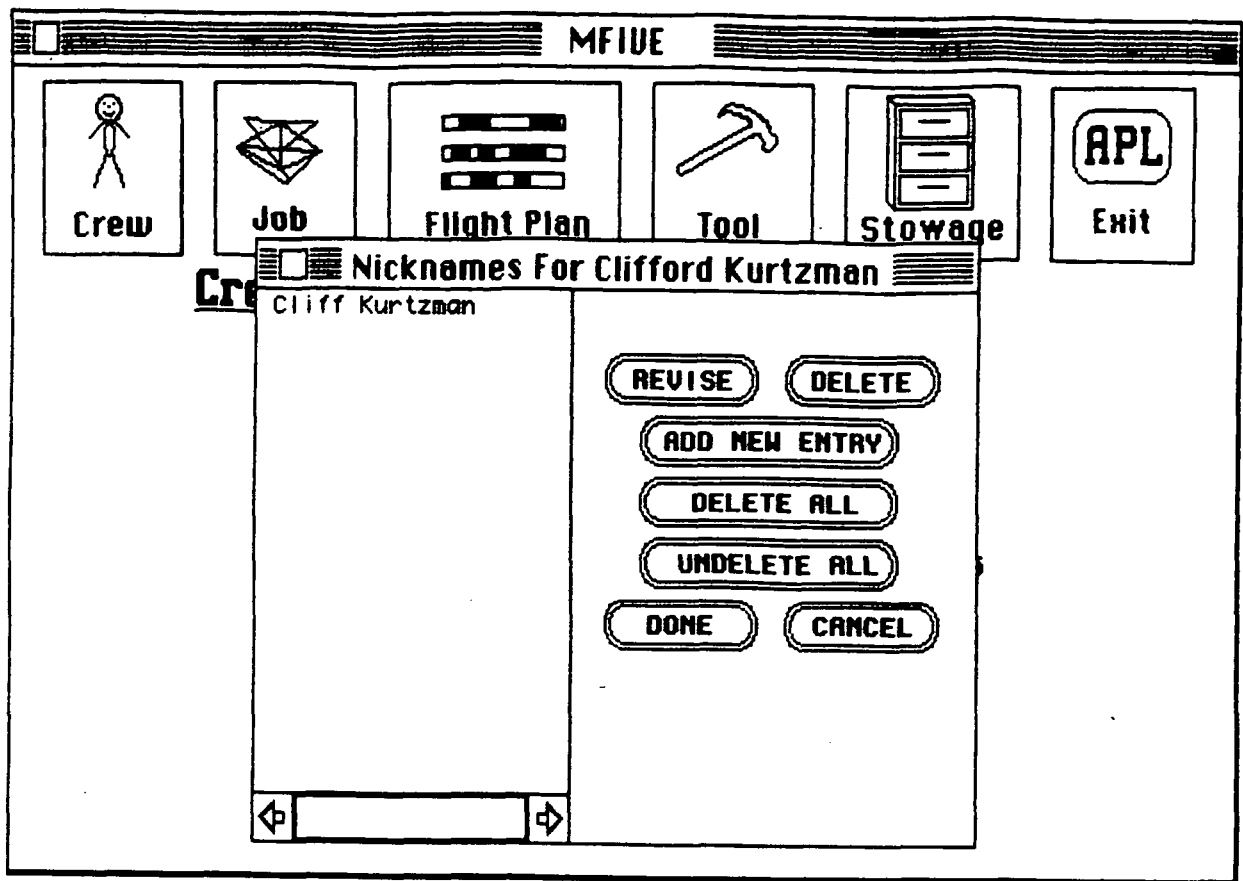


Figure 13

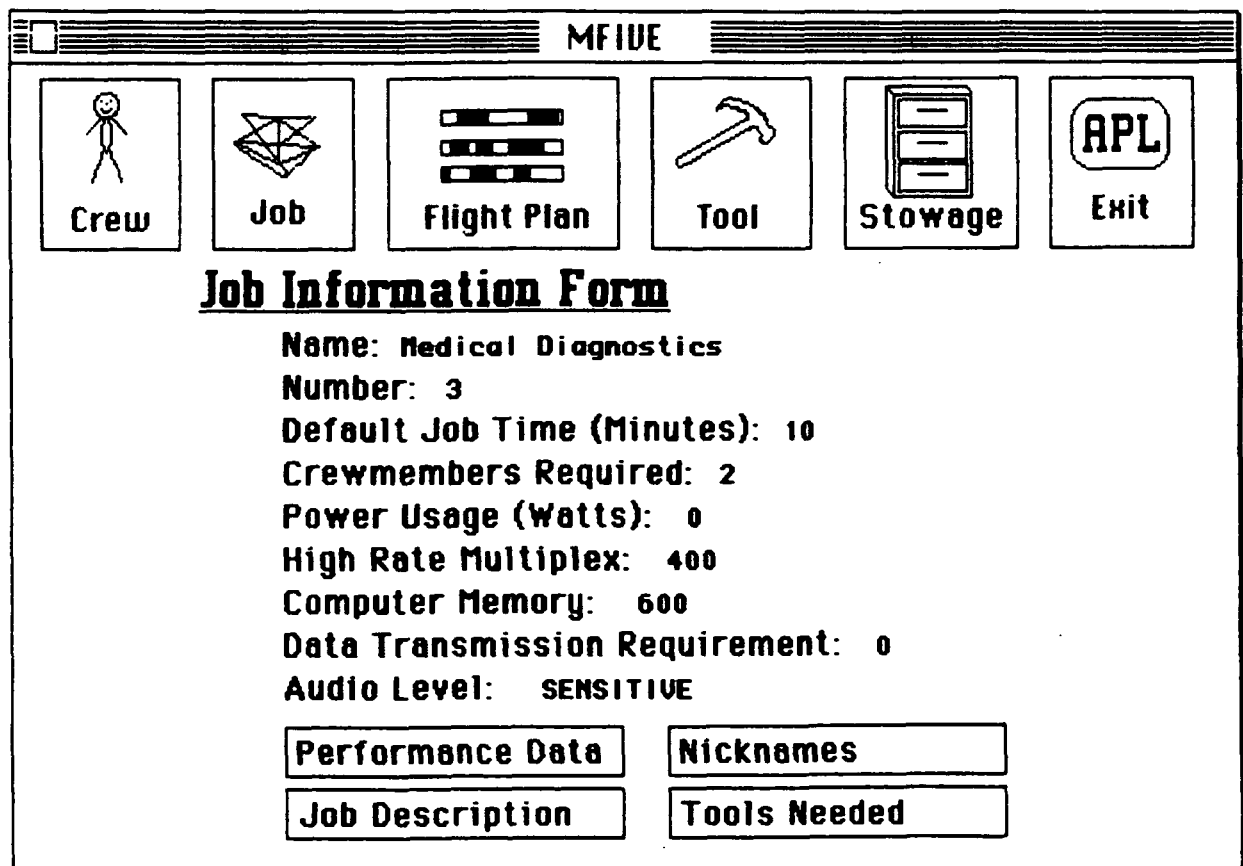


Figure 14

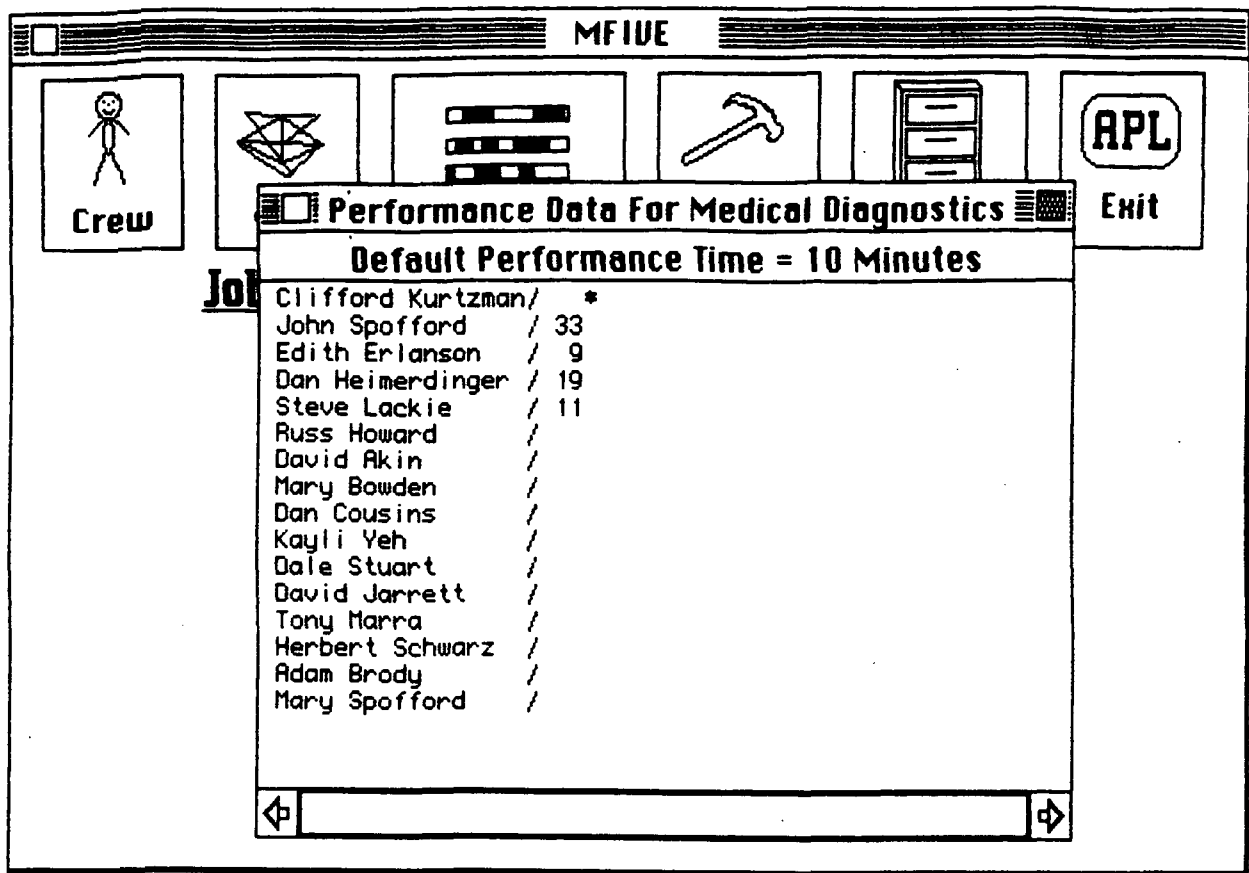


Figure 15

information that can be accessed from the PERFORMANCE DATA box on the Crewmember Information Form, but indexed by job instead of by crewmember). In general, each job has a default performance time which is displayed at the top of the window (e.g., 10 minutes in Figure 15). Only performance times for crewmembers who have other than the default value are explicitly shown (see Figure 15). Clicking on a particular crewmember allows the user to change the performance time from the default value or delete the performance time. A crewmember for which the performance time has been deleted (this is indicated by an * in Figure 15) cannot be assigned that task by the scheduler. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The NICKNAMES box: Clicking this box displays a scrollable window (similar to Figure 13) which contains acceptable alternate names for the job. By clicking on the REVISE button and then on a nickname, one can make revisions. Deletions are made by clicking on the DELETE button and then on the nickname to be deleted. The other buttons are self explanatory. The white box at the upper left corner of the window is the same as the DONE button.

The JOB DESCRIPTION box: Clicking this box displays a scrollable and resizable window containing information (such as instructions) pertinent to the job (Figure 16). Clicking within the box moves the cursor to allow editing and addition of text. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The TOOLS NEEDED box: Clicking this box displays a scrollable window showing which tools (and how many of them) are needed to perform the job (Figure 17). Clicking on a particular tool allows the user to change the quantity used, or to delete the information previously entered. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The Flight Plan Information Form (Figure 18)

The NAME slot: Same as for the Crewmember Information Form.

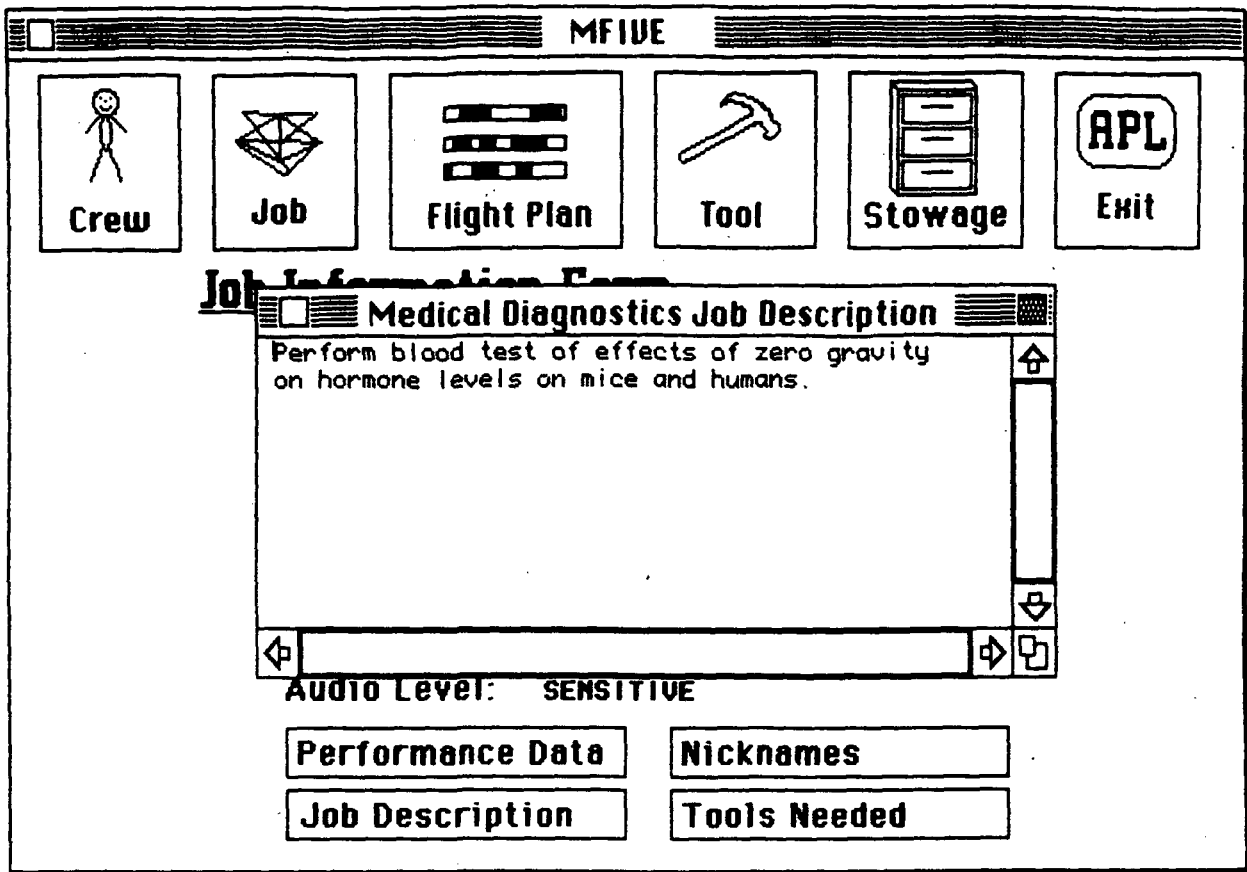


Figure 16

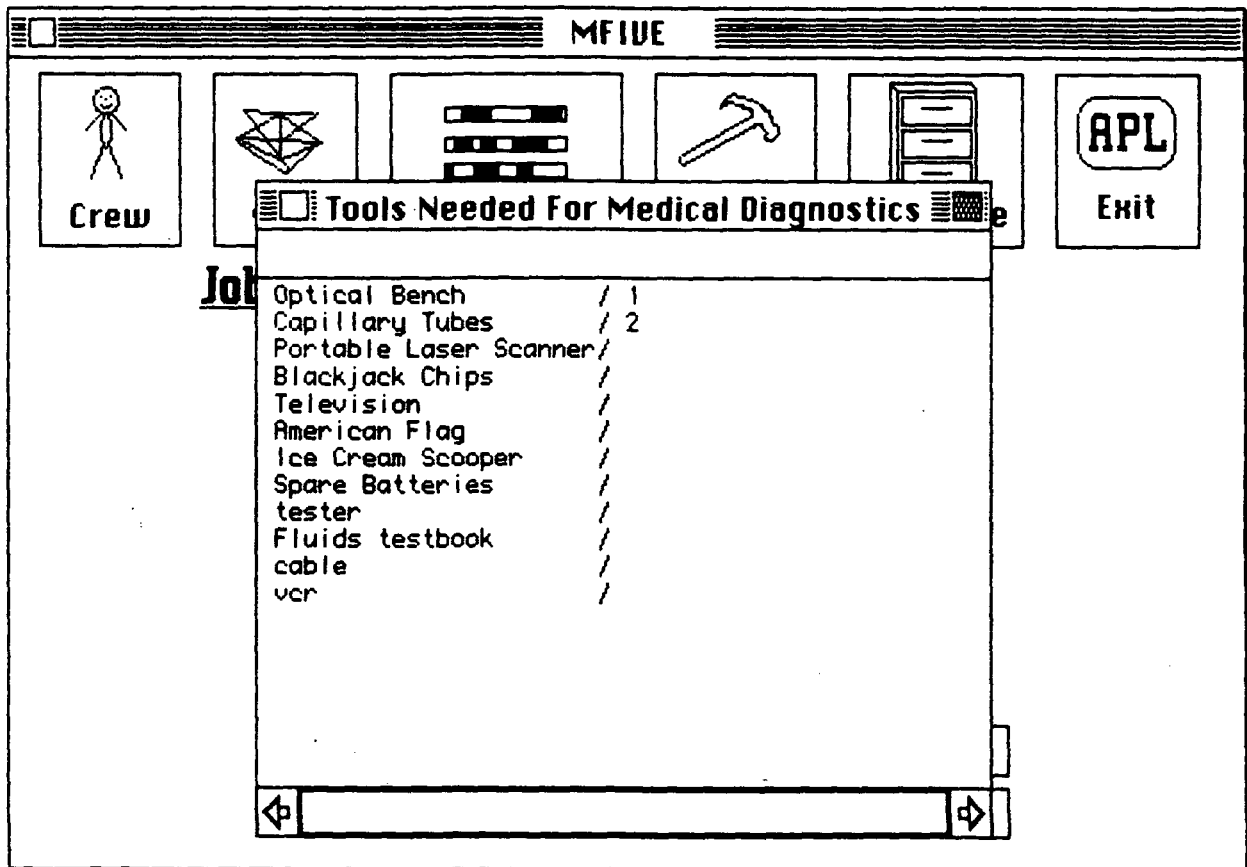








Figure 17

MFIDE

 Crew	 Job	 Flight Plan	 Tool	 Stowage	 Exit
---	--	--	---	--	---

Flight Plan Information Form

Name: Demo
Number: 3
Start Date: AUG 30, 1988 12:00
End Date: AUG 31, 1988 2:00
Maximum Power Usage: 3000
Maximum Multiplex Usage: 2000
Maximum Memory Usage: 2000
Maximum Data Transmission: 3000

Assigned Crewmembers

Assigned Jobs

Schedule

Figure 18

The NUMBER slot: Same as for the Crewmember Information Form.

The START DATE and END DATE slots: These are the earliest and latest dates at which a job can be scheduled for this flight plan.

The MAXIMUM POWER USAGE, MAXIMUM MULTIPLEX USAGE, MAXIMUM MEMORY USAGE, AND MAXIMUM DATA TRANSMISSION slots: These slots specify the maximum amount of these resources that can be used at any time during the flight plan.

The ASSIGNED CREWMEMBERS box: Clicking this box displays a window showing a list of the crewmembers and highlighting those selected for this flight plan (Figure 19). Clicking on particular crewmembers will add or remove them from the flight plan. A maximum of eight crewmembers can be selected for any flight plan.

The ASSIGNED JOBS box: Clicking this box displays a window showing a list of the jobs and highlighting those selected for this flight plan (Figure 20). Clicking on particular jobs will add or remove them from the flight plan.

The SCHEDULE button: After a complete flight plan has been determined (i.e., crewmembers, jobs and other information completed), clicking on the SCHEDULE button will transfer the user to the scheduler to assemble a timeline meeting all requirements and constraints (See Section 4.2.2).

The Tool Information Form (Figure 21)

The NAME slot: Same as for the Crewmember Information Form.

The NUMBER slot: Same as for the Crewmember Information Form.

The QUANTITY IN STOCK slot: This specifies how many copies of the tool are available. This information is not modifiable by the user, but is instead determined by the number of copies of the tool entered via the DEFAULT LOCATION AND STATUS box, as described below.

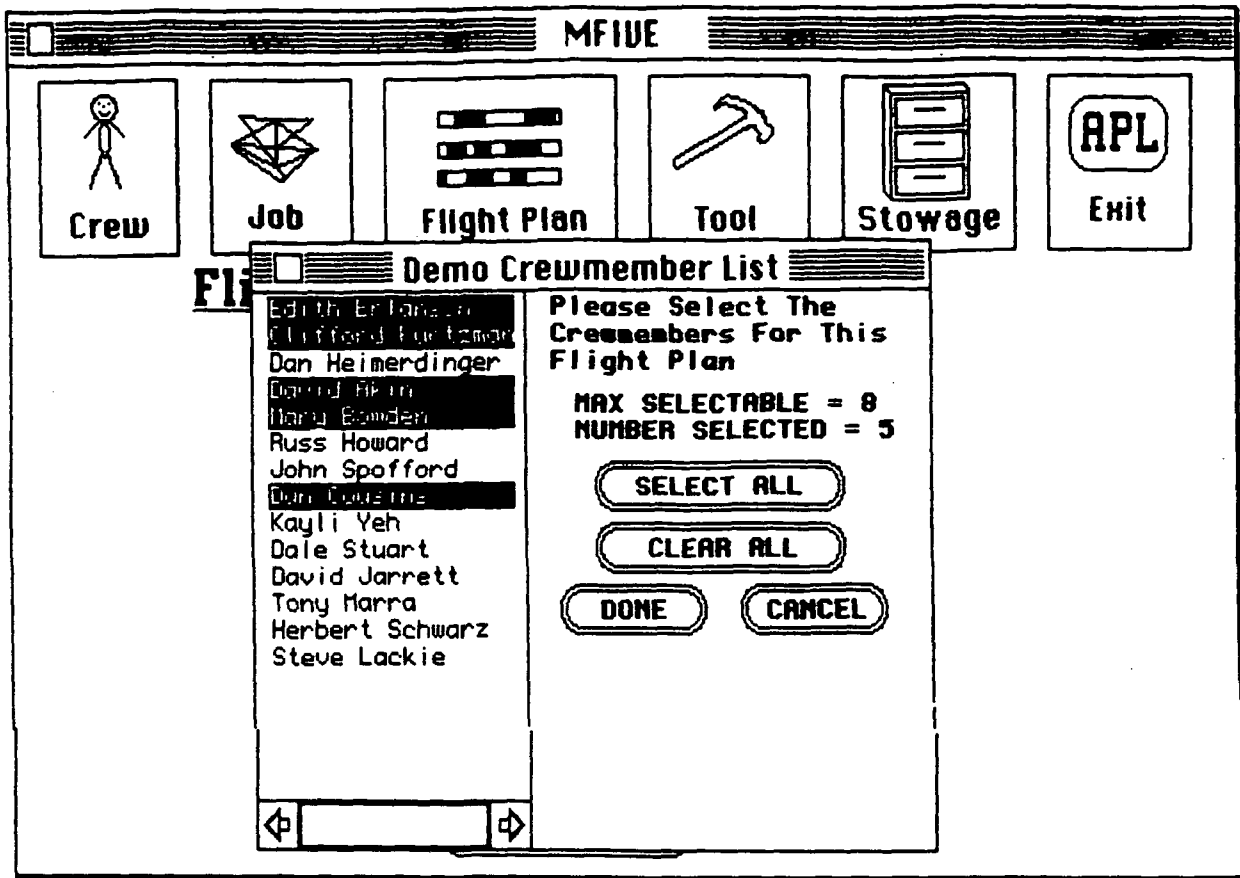


Figure 19

ORIGINAL PAGE IS OF POOR QUALITY

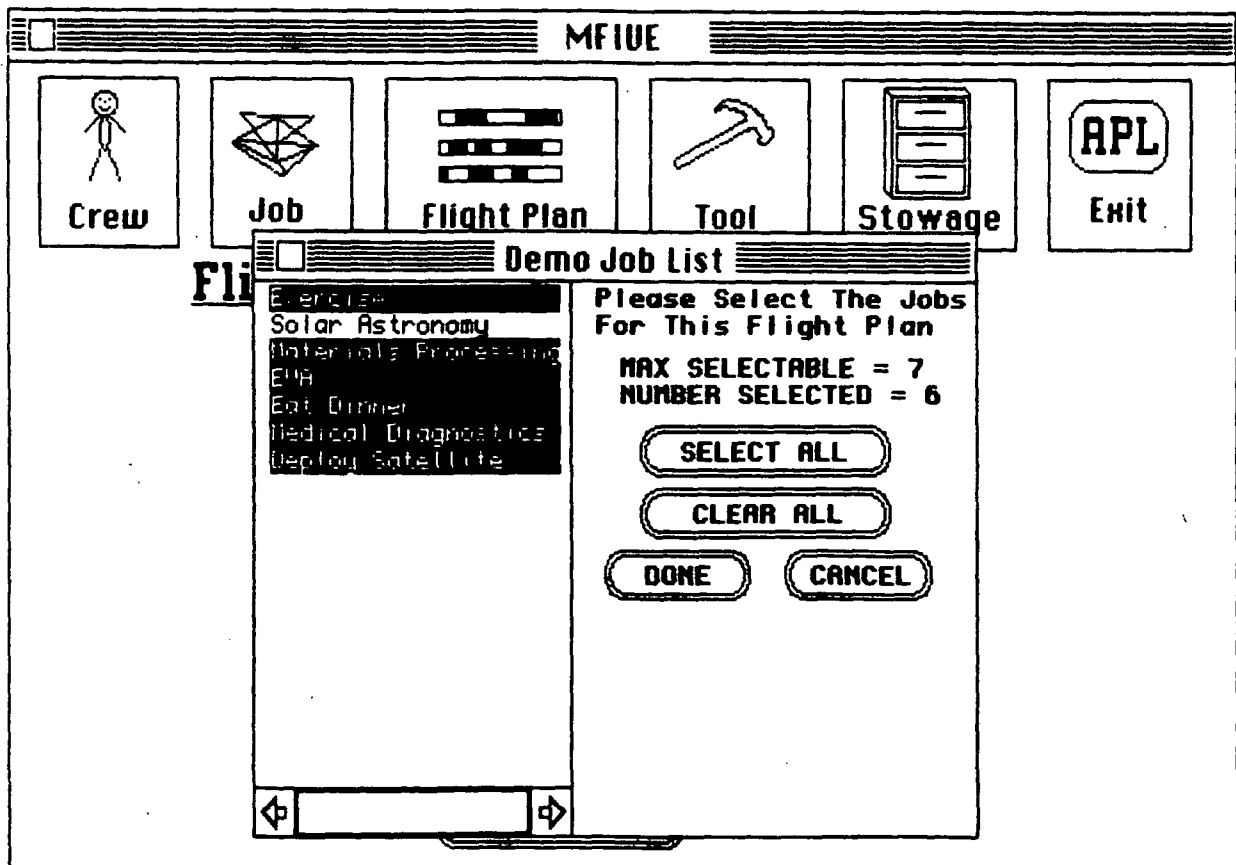


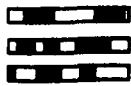





Figure 20

MFIVE

 Crew	 Job	 Flight Plan	 Tool	 Stowage	 Exit
--	---	---	--	---	--

Tool Information Form

Name: cable
Number: 11
Quantity in Stock: 3
Effective Volume (liters): 10

**Default Location
and Status**

Usage Instructions

Job Applications

Search

Figure 21

The **EFFECTIVE VOLUME** slot: This specifies the volume that the tool takes up. In initial versions of MFIVE, any tool can be stored in any compartment which has a larger effective volume. More sophisticated versions will take into account the dimensions of the tool and the compartment.

The **DEFAULT LOCATIONS AND STATUS** box: This identifies the expected stowage location(s) and status of each copy of the tool. Clicking this box displays a window showing a list of the nominal stowage location and status for each copy of this tool (Figure 22). Copies of a tool can be deleted by clicking on the copy number. Default locations can be revised by clicking on the current default location and responding to a prompt for the new default location (one can also type "help" for a list of possible options). Selected stowage locations must have an effective volume greater than that of the tool. Tool status can similarly be revised by clicking on the current tool status and responding to a prompt for the new tool status (or by typing "help" for a list of possible options). New copies of a tool can be added by clicking on **CREATE NEW COPY** and then responding to the prompts for default location and tool status.

The **USAGE INSTRUCTIONS** box: Clicking this box displays a scrollable and resizable window containing information (such as instructions) pertinent to the using this tool (Figure 23). Clicking within the box moves the cursor to allow editing and addition of text. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The **JOB APPLICATIONS** box: Clicking this box displays a scrollable window showing the jobs for which this tool is used (Figure 24). In the example shown, we see that the job **WATCH TV** uses one of the three cables, while none of the other jobs require the cable. Clicking on a particular job allows the user to change the quantity used, or to delete the information previously entered. Clicking the white box at the upper left corner of the window closes the window and adds any changes to the data base. Clicking on the grey box in the upper right corner of the window closes the window, but cancels any changes made.

The **SEARCH** button: Clicking this starts the searcher to produce a ranking of the most likely places to find the tool (See Section 4.2.3).

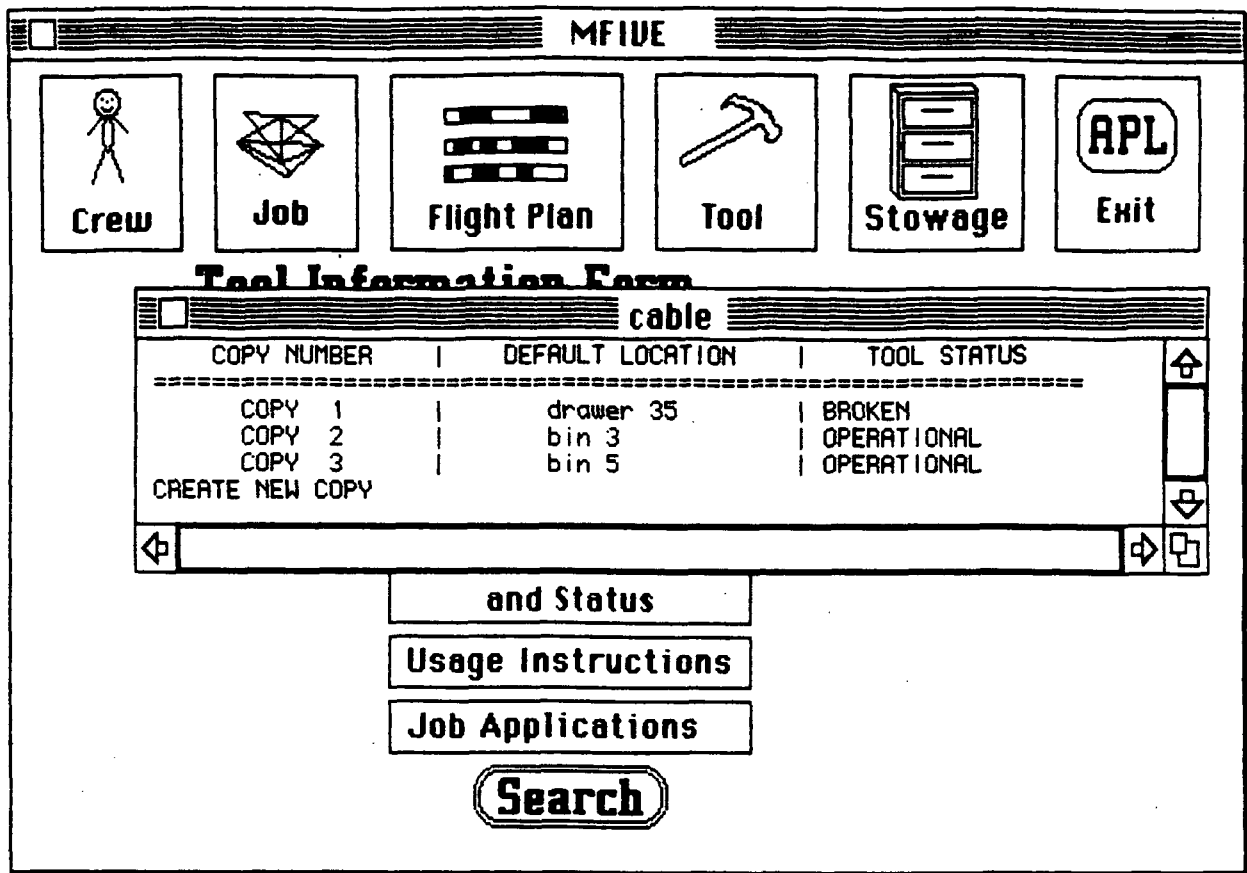


Figure 22

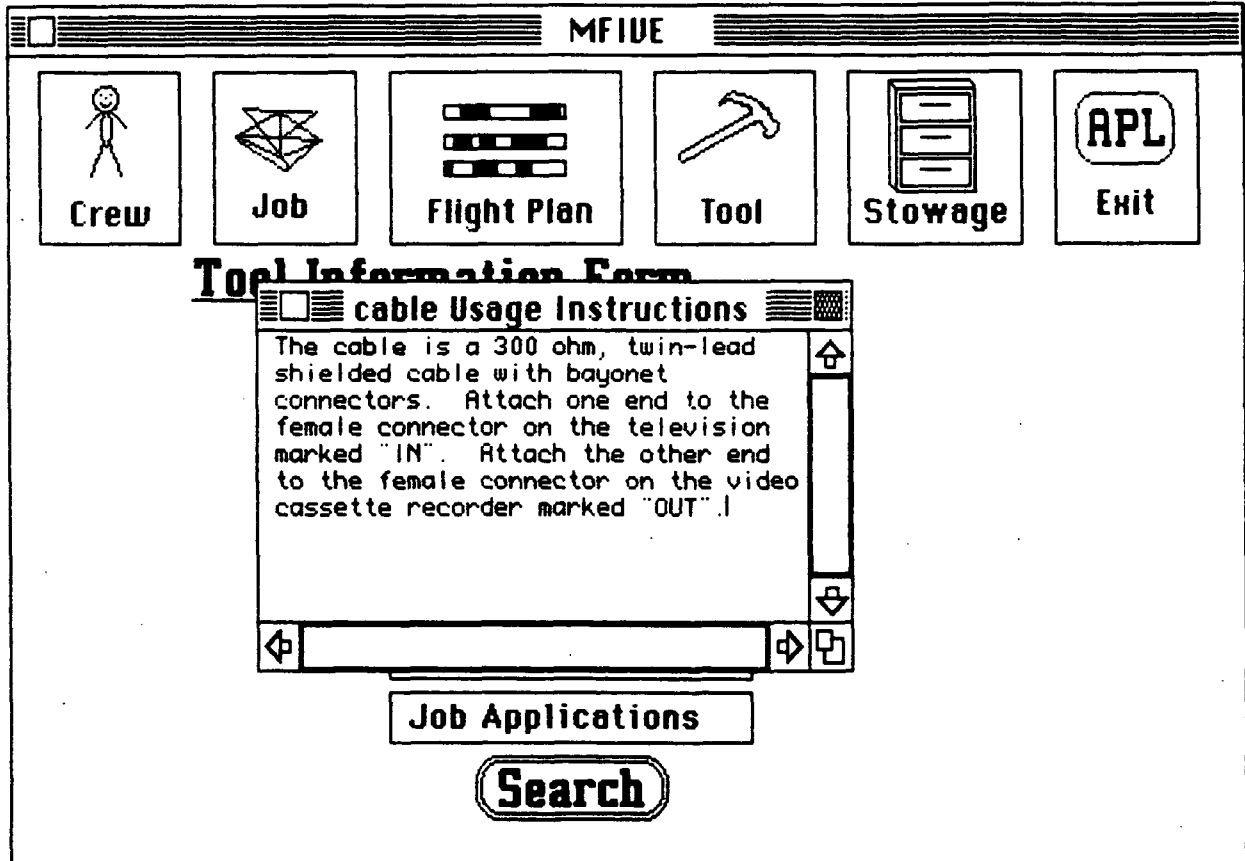


Figure 23

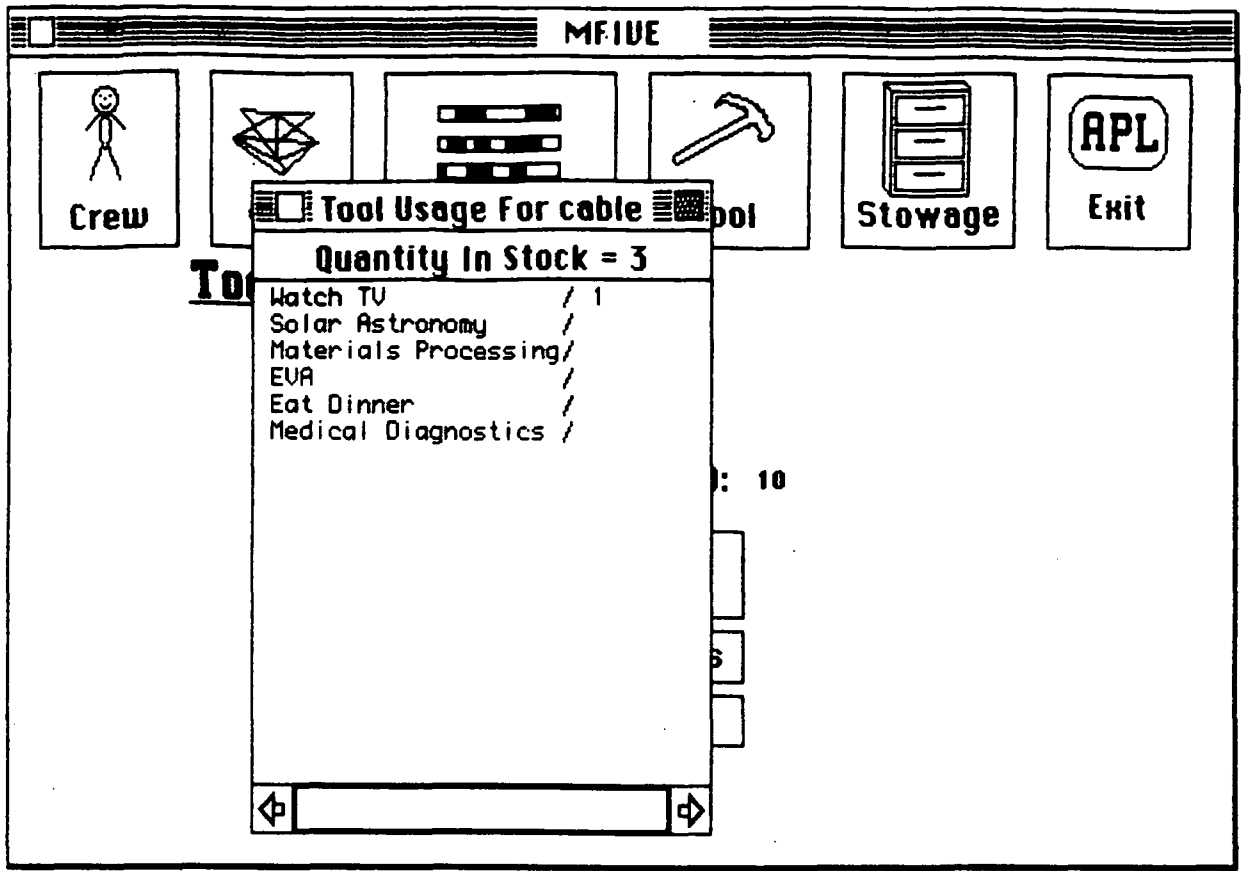


Figure 24

The Stowage Information Form (Figure 25)

The COMPARTMENT NAME slot: Same as for the name slot on the Crewmember Information Form.

The NUMBER slot: Same as for the Crewmember Information Form.

The EFFECTIVE VOLUME slot: This specifies the volume of the compartment. In initial versions of MFIVE, any tool can be stored in any compartment which has a larger effective volume. More sophisticated versions will take into account the dimensions of the tool and the compartment.

The MODULE NUMBER slot: This specifies the module in which the compartment is located.

The X LOCATION slot: This specifies the X location of the compartment (referenced from one end of the module).

The THETA LOCATION slot: This specifies the angle (in degrees) of the compartment, referenced from the defined "vertical".

The DEFAULT CONTENTS box: This identifies the expected tools in this stowage compartment and their statuses (Figure 26). This window is not modifiable. To add or delete a tool from a compartment or to change its status, the user should call up the particular tool's information form and make the appropriate revisions via the DEFAULT CONTENTS AND STATUS box.

The OPTIONS Menu

The OPTIONS menu at the top of the screen (Figure 27) contains entries to enable the user to save, load, or delete an entire data base (e.g. all crew, job, flight plan, tool, and stowage parameters). This allows a user to plan and compare multiple scenarios.

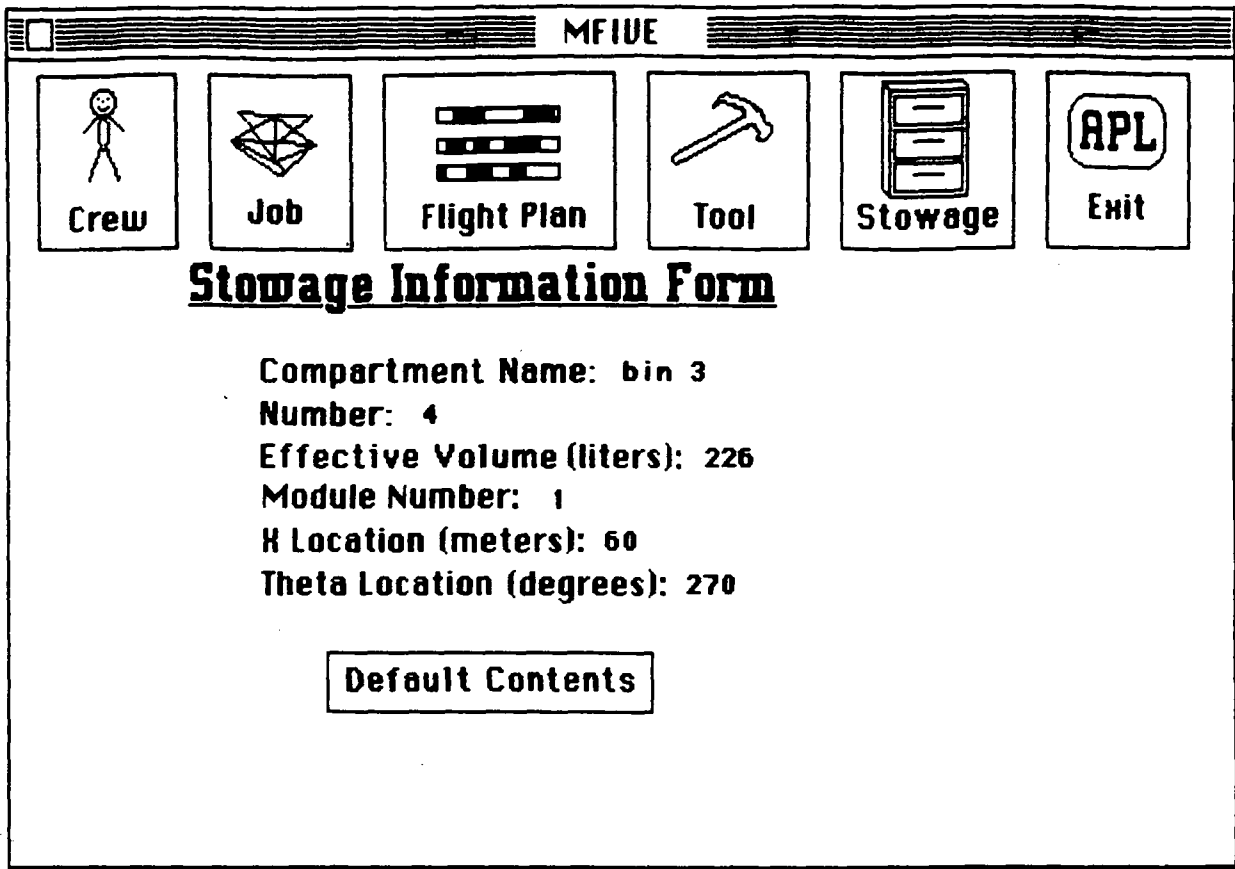


Figure 25

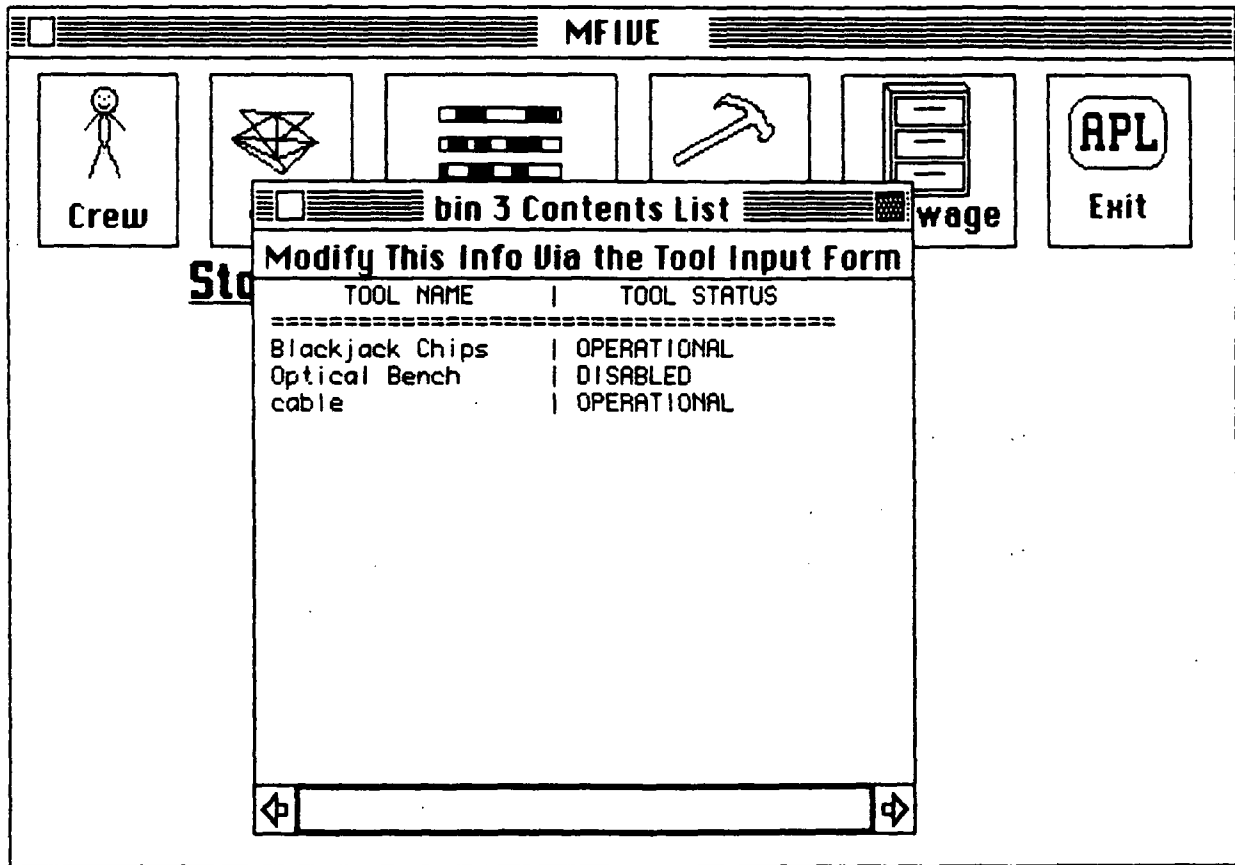


Figure 26

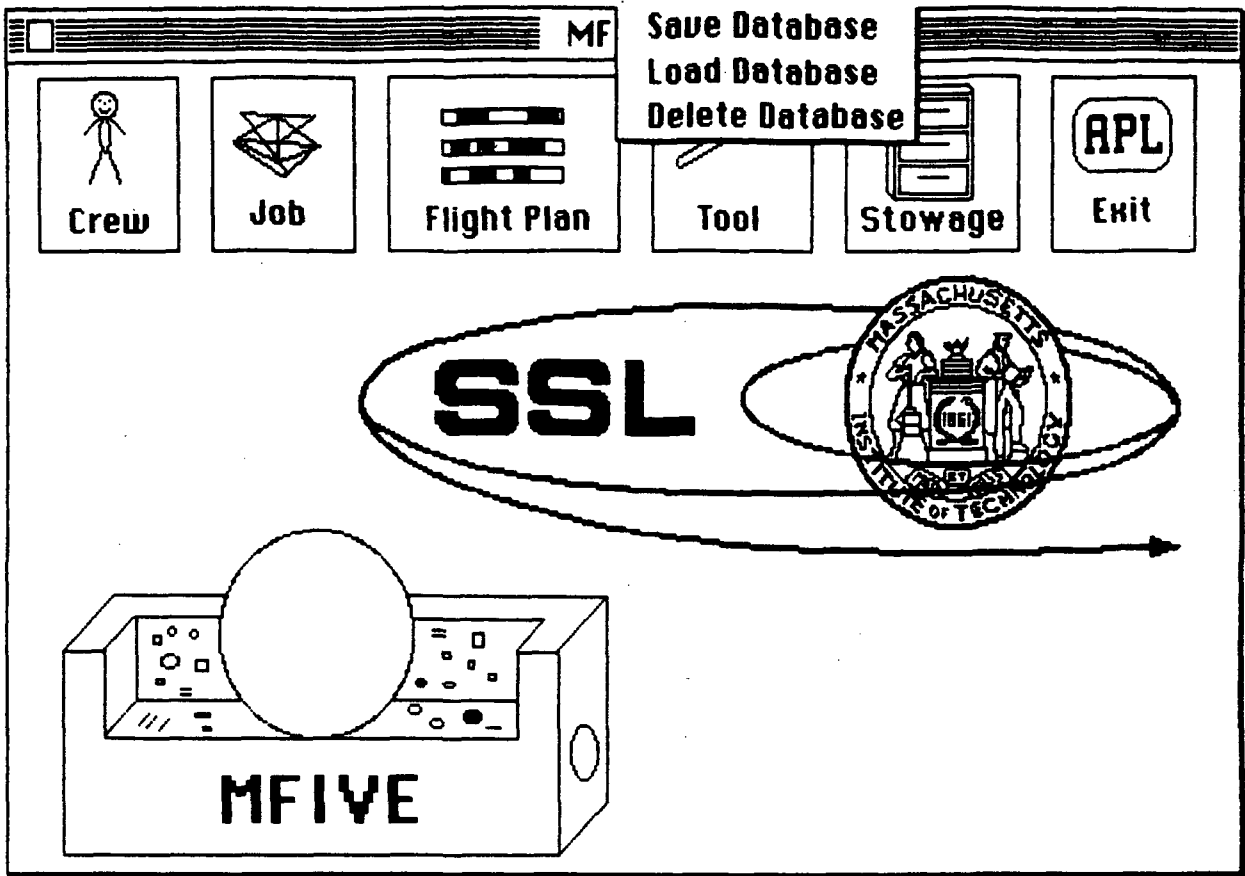


Figure 27

4.2.2 The Scheduler

Clicking the SCHEDULE button on the flight plan information form (Figure 18) starts the scheduler working and presents the user with a scheduling worksheet (Figure 28). The scheduling worksheet enables a user to select jobs for scheduling, set time and target constraints, inspect resource usage, and allows manual and automatic task scheduling.

The Scheduling Worksheet

Central to the scheduling worksheet are the activity timelines of each of the crewmembers assigned to the flight plan. Figure 29 shows a flight plan midway through the scheduling process, where some of the jobs have already been scheduled. The activity timelines are the long horizontal rectangles in the upper part of the figure. For this flight plan, there are timelines for five crewmembers, denoted C1 through C5. The names of each of these crewmembers appears in the lower right corner of the figure.

Beneath the crewmembers timelines is a line of text stating the start and end dates for this flight plan. These are followed, in parentheses, by the duration of the flight plan, in the format DAYS/HOURS:MINUTES. At the top of the worksheet is a ruler showing time divisions along the timeline (in the format DAYS:HOURS). For example, the ruler in Figure 29 starts at zero days and zero hours (0/00) from the start of the timeline (August 30, 1988 at 12:00) and ends at zero days and six hours (0/06) into the flight plan (i.e., August 30, 1988 at 18:00).

The gray shaded rectangles inside the activity timelines correspond to jobs which have been assigned to the crewmembers. For example, in Figure 29 we see that Job 3 (denoted J3) has been assigned to crewmember 2 (Clifford Kurtzman) from 0/3:55 (i.e., zero days, three hours, and fifty five minutes into the flight plan) till 0/4:35. If we wish to find out what job J3 corresponds to, we can click the mouse inside its shaded rectangle and a box will pop up (Figure 30) explaining that J3 corresponds to Job 3, EVA. In addition to the presentation of this explanation box, the exact hour and minute at which the mouse is centered is displayed in the box at the upper right corner (i.e., 3:55 in Figure 30). The user can thus clearly delineate the start and end times of a task by sliding the mouse from the beginning to end of the shaded rectangle.

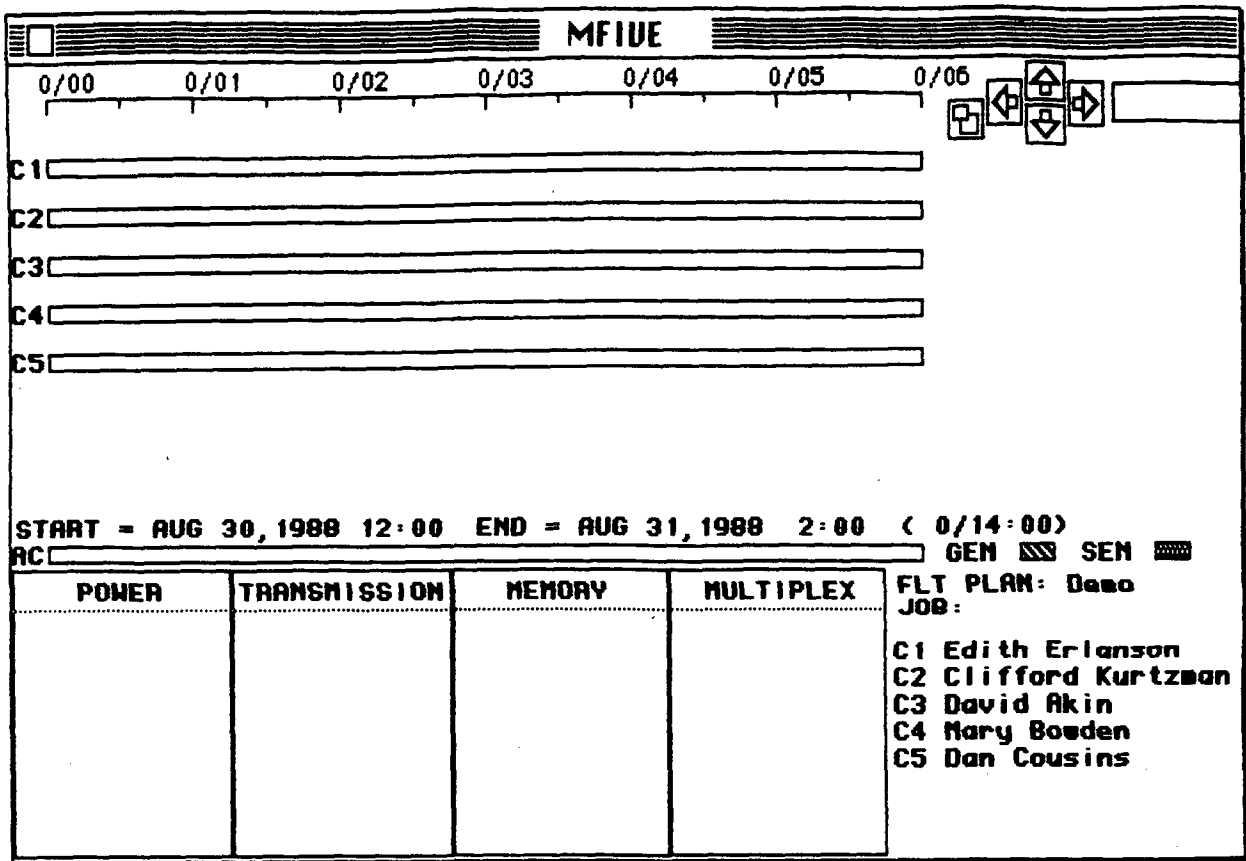


Figure 28

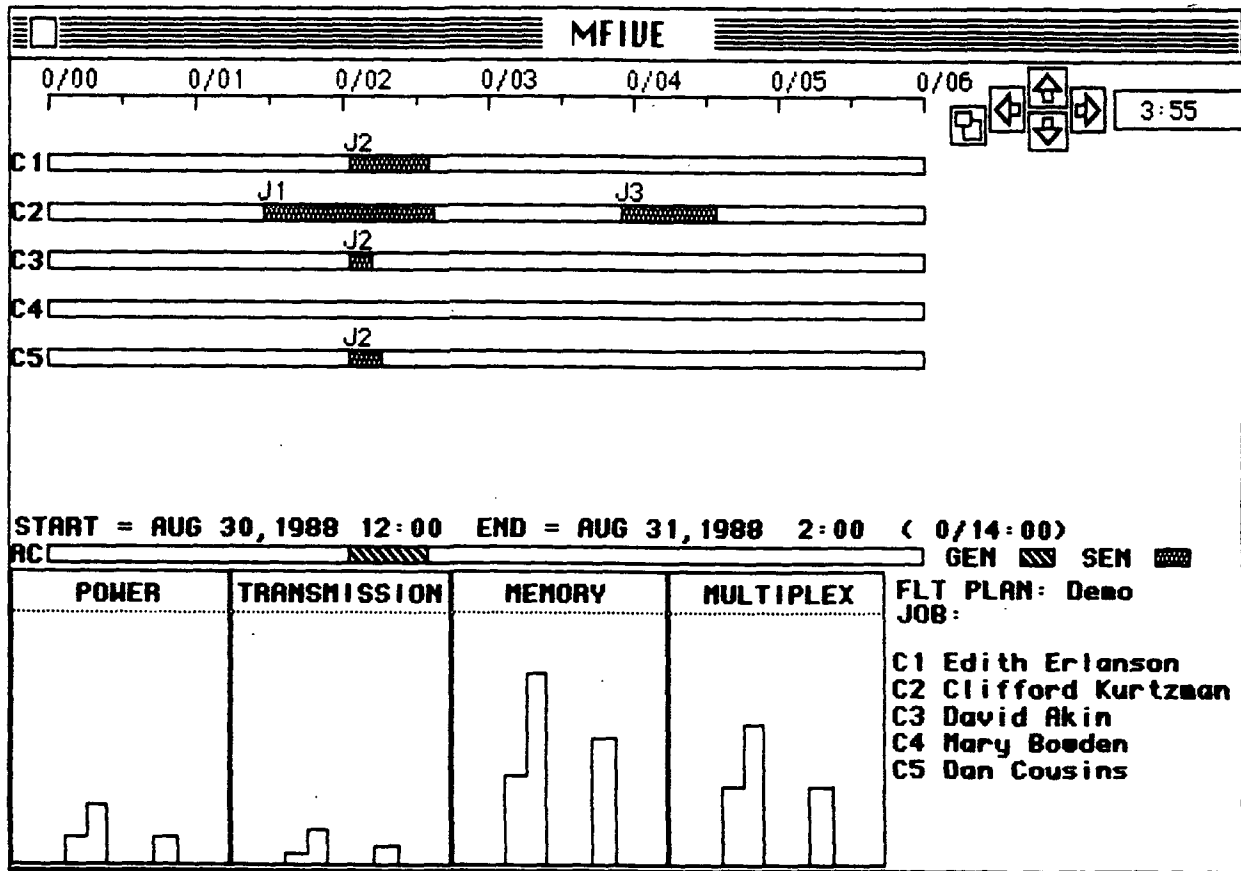


Figure 29

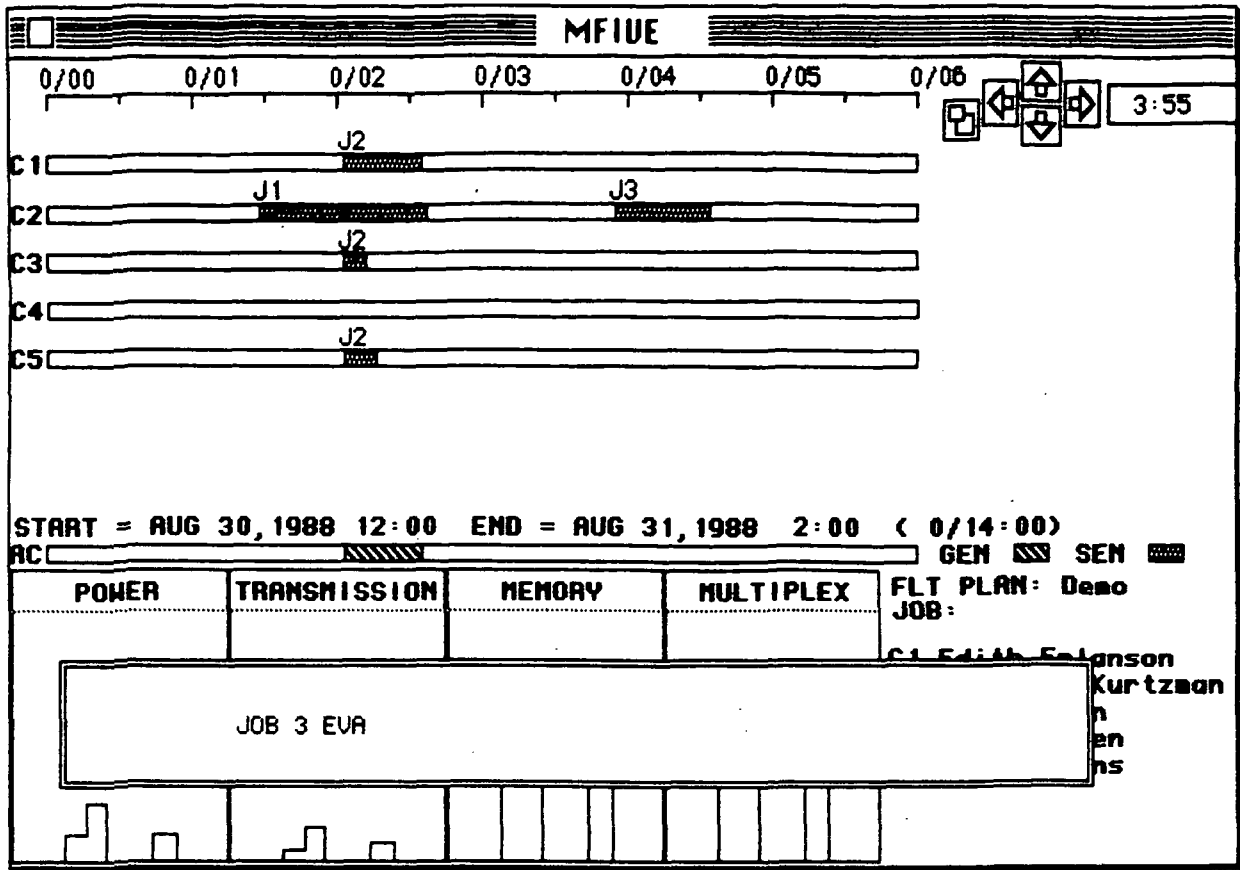


Figure 30

The scale and origin of the timeline ruler at the top of the scheduling worksheet can be altered via the arrows at the upper right. At the start of the scheduler, the ruler is set to start at the beginning of the flight plan, and to show the first six hours. This corresponds to displaying one minute of flight plan for each pixel on the Macintosh screen. Clicking the up or down arrows either adds or subtracts one minute per pixel to the displayed timeline. For example, clicking the up arrow in Figure 29 results in Figure 31, in which the first twelve hours of the flight plan are displayed. Note that at this two minutes per pixel setting the rectangles corresponding to the scheduled jobs are only half as wide.

Clicking the right or left arrow shifts the origin of the timeline. For example, clicking the right arrow in Figure 29 results in Figure 32, in which the timelines start at 0/04 and end at 0/10. In general, these arrows shift the timeline over by two thirds of its current width (i.e., $\frac{2}{3} \times 6 \text{ hours} = 4 \text{ hours}$: therefore the timeline was shifted over 4 hours).

Clicking on the box with the two rectangles inside of it (next to the arrows) presents the user with a window in which the number of minutes per pixel and the origin of the timeline can be explicitly set (Figure 33).

The four large boxes at the bottom of the scheduling worksheet display the resource usage for the timeline. The width that each of these boxes displays corresponds to the width of the scheduling rectangles (i.e., from 0/00 till 0/06 in Figure 29). Changing the width or origin of the ruler changes the resource displays as well (e.g., see Figure 31 and Figure 32). The height of the resource boxes are scaled so that the dashed line near the top indicates usage at the maximum specified for the flight plan (see Figure 18). By clicking on a resource box, the user can call up a window displaying the usage in expanded form, and with the axes labelled (Figure 34).

Directly above the four resource boxes, is a horizontal rectangle (labelled AC) which displays the audio constraint. Times at which a noise generating activity is being performed are signified by striped rectangles, and times at which a noise sensitive activity is being performed are signified by gray rectangles. In Figure 29 it can be seen that there is a noise generating rectangle corresponding with the duration of job 2.

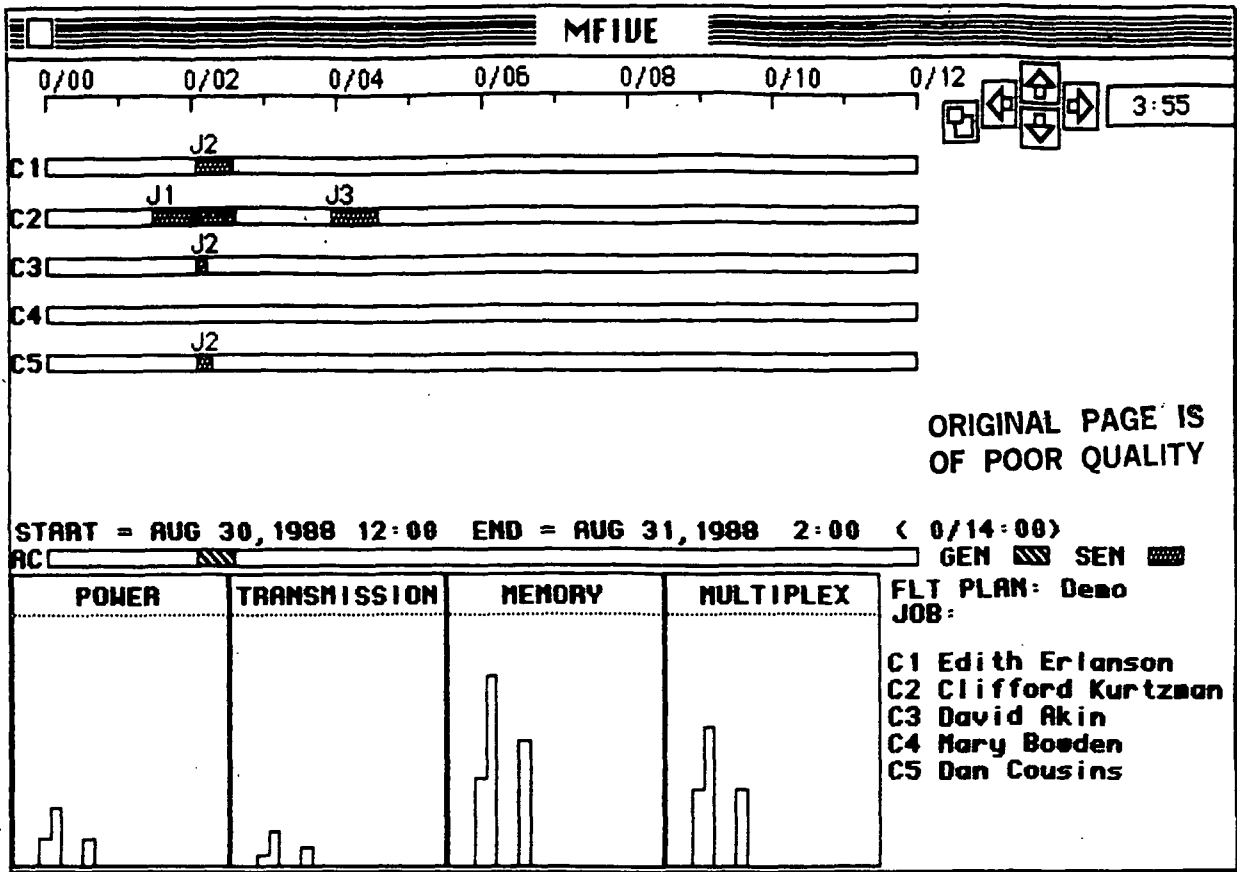


Figure 31

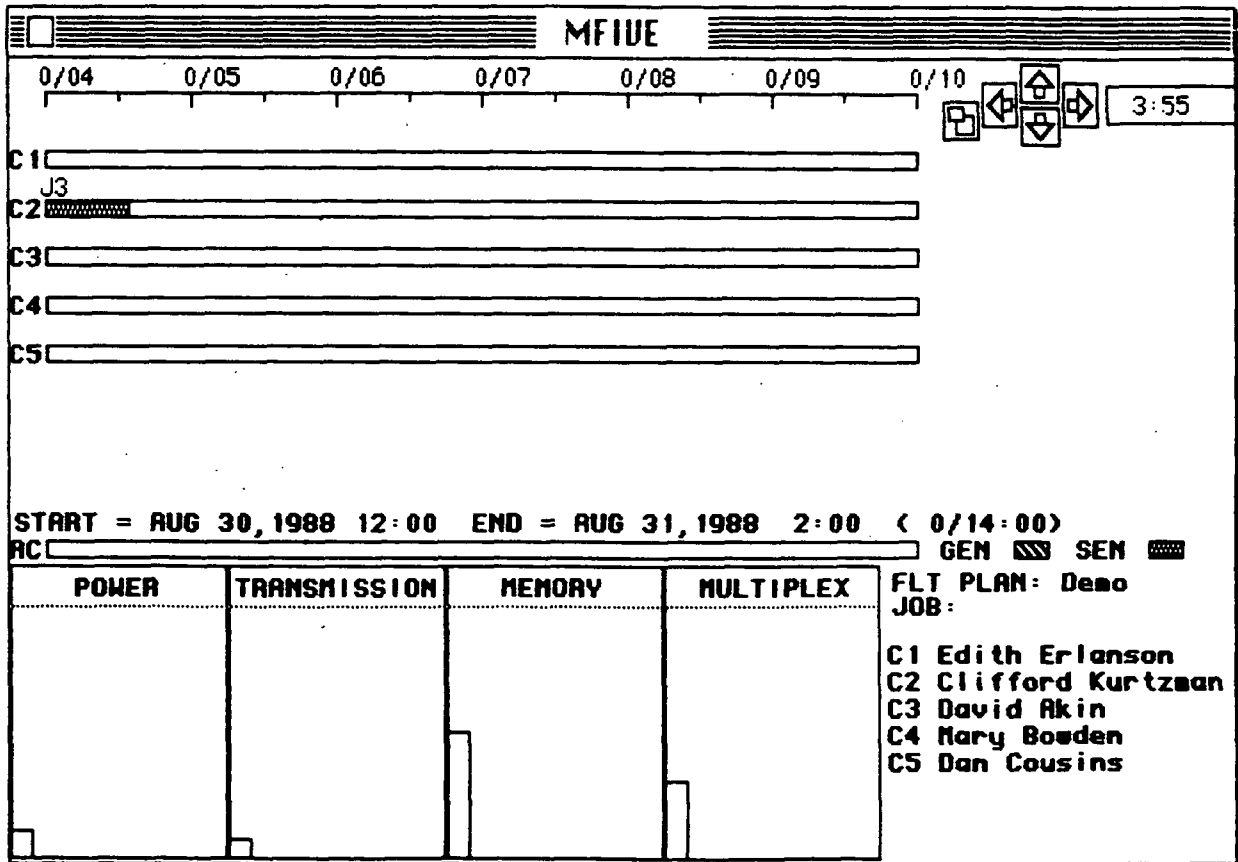


Figure 32

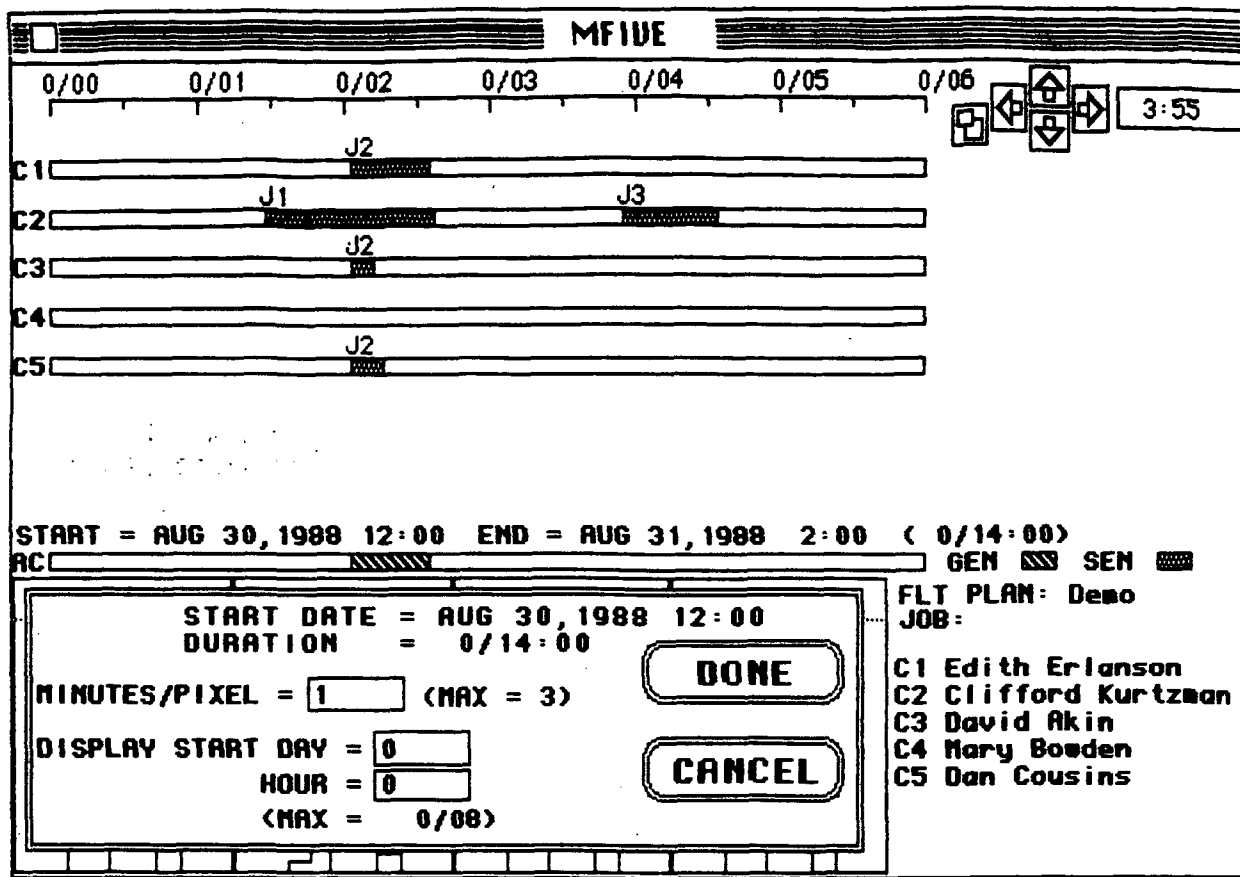


Figure 33

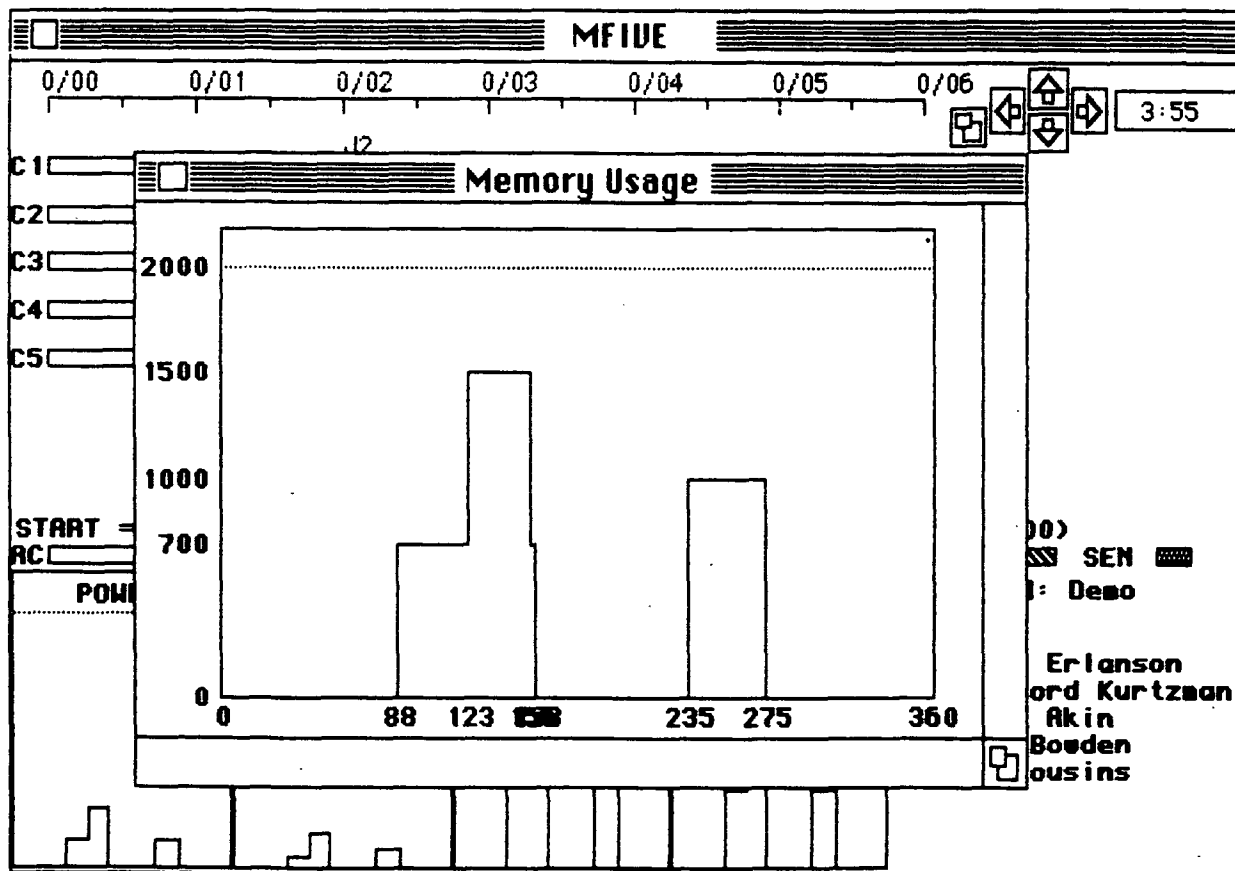


Figure 34

Selecting a Job to be Scheduled

There are three ways to select a job for scheduling. Figure 35 shows the **OPTIONS** menu, at the top of the Macintosh screen. Choosing the **SELECT JOB** option from this menu will instruct **MFIVE** to pick the next job to be scheduled, based upon its internal heuristic. Choosing the **AUTO JOB SELECT** option has the same effect, except that after the job which **MFIVE** selects is scheduled, **MFIVE** will automatically pick another job for scheduling. Finally, the user can determine the next job to be scheduled by selecting the option **DISPLAY JOB INFO**. This will present the information window shown in Figure 36. This window shows, for each of the jobs, the number of crewmembers required, the default time, the minimum time for any crewmember, the resource usages, and the audio status of each job. An asterisk next to the name of a job indicates that it has already been scheduled.

Clicking on the name of a job selects that job for scheduling. Figure 37 shows how the scheduling worksheet looks after **Deploy Satellite (Job 4)** has been selected, and the job information window has been closed. Immediately above the names of the crewmembers appears the name of the job being scheduled, and below the right arrow at the top appears the number of crewmembers necessary to perform this job.

Immediately to the right of the activity timelines appears the number of minutes it takes each of the crewmembers to perform this job. For example, Figure 37 shows that crewmembers 1 and 4 take 30 minutes to perform Job 4, while crewmember 2 takes 40 minutes, crewmember 3 takes 11 minutes, and crewmember 4 is not rated for this job. Next to the numeric performance times are rectangles which have a width corresponding to the crewmembers performance time. To the right of the rectangles are numbers indicating the total workload (in minutes) for which each of the crewmember have already been scheduled. For example, in Figure 37 crewmember 2 has been assigned jobs 1 and 3, for a total of 110 minutes of worktime.

When a job has been selected for scheduling, parts of the crewmembers activity timelines may become "blacked out", as in Figure 37. This is because these time slots are infeasible due to one or more constraint violations. To find the reason an interval is blacked out, the user can simply click the mouse on the blackened region, and a window will appear with an explanation (Figure 38). In the case of Figure 37, the blackened intervals are due to the fact that scheduling Job 4 in these intervals would exceed the memory resource limit.

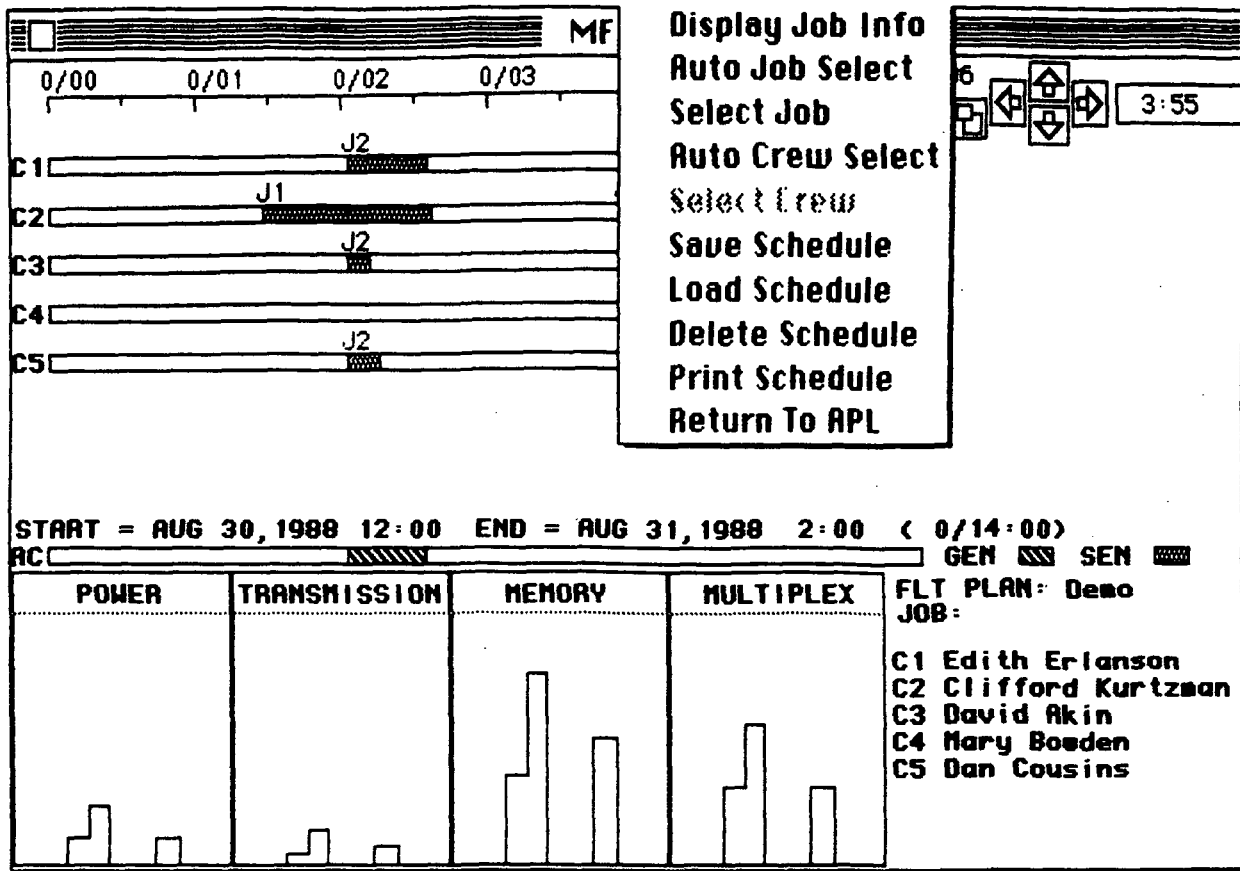


Figure 35

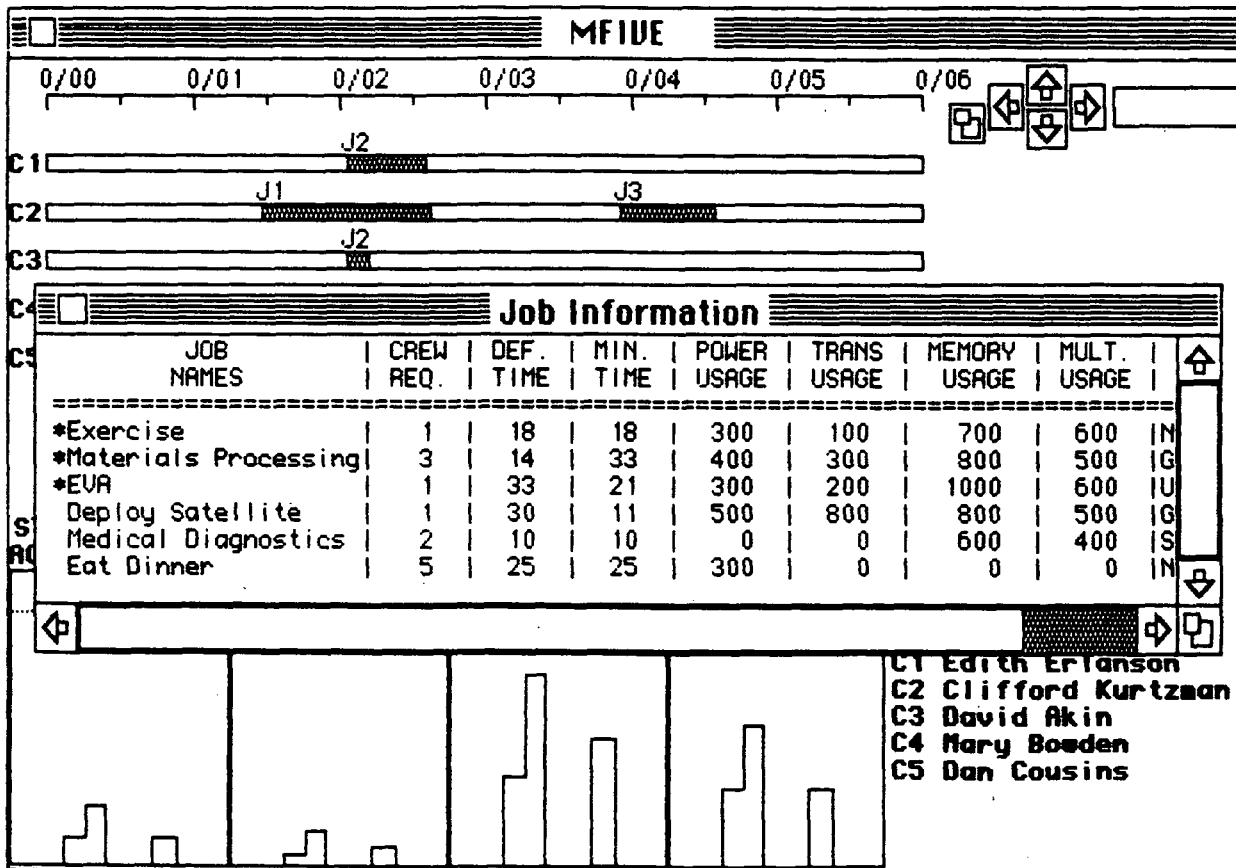


Figure 36

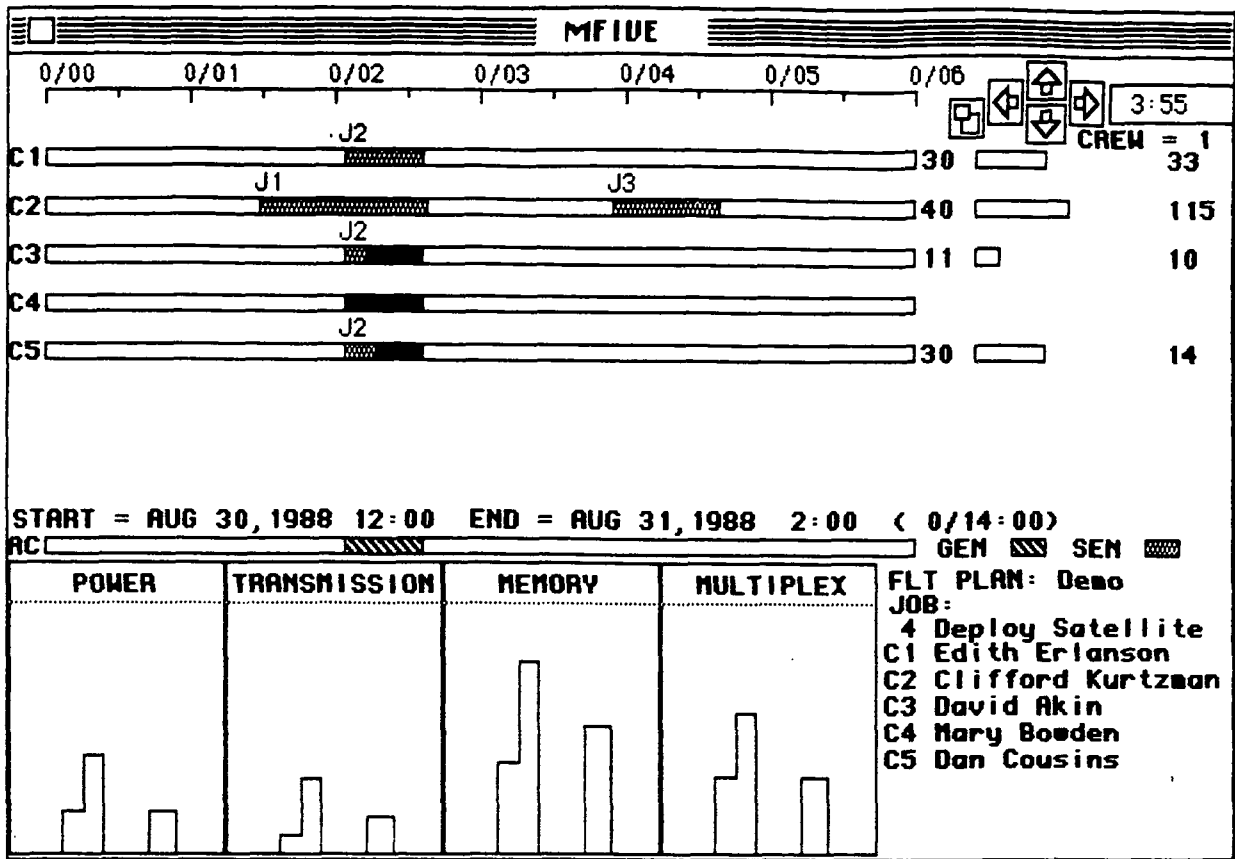


Figure 37

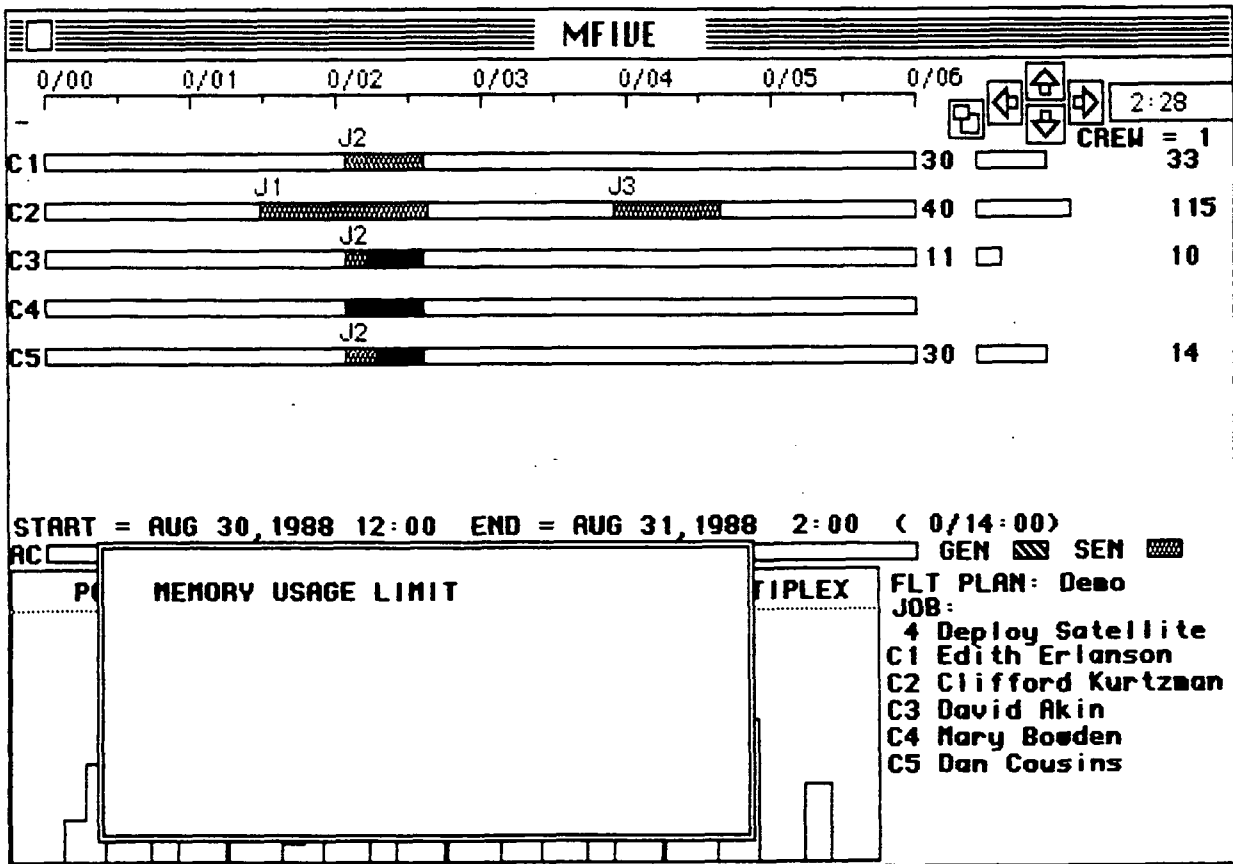


Figure 38

Once a job has been selected for scheduling, a crewmember may be assigned by several methods. The user can click on the crewmember's rectangle (to the right of the activity timeline) and slide this rectangle (with the mouse) to the desired spot on the crewmembers timeline. The exact time at which the rectangle is positioned is always displayed at the upper right of the screen, so that the rectangle can be accurately positioned. Figure 39 shows the result of sliding crewmember 5's rectangle to the 3:37 position. The resource usages are appropriately updated.

In order to assign a job to a crewmember at the crewmembers earliest available start time, the user clicks the mouse on the crewmember's number (e.g., "C5", at the very left of crewmember 5's timeline). Figure 40 shows the result of clicking on C5.

Crewmember selection can be completely automated by choosing the SELECT CREW option from the OPTIONS menu. MFIVE will then choose the best crewmember based upon its internal heuristic, and assign the job to that crewmember at the crewmember's earliest available time (Figure 41). The internal heuristic assigns the task to the crewmember who will have the least total workload, in this case crewmember 3.

Choosing the AUTO CREW SELECT option from the OPTIONS menu will have the same effect as the SELECT CREW option, but then after every subsequent job is selected for scheduling, the crewmember will automatically be assigned. By selecting both the AUTO CREW SELECT option and the AUTO JOB SELECT option, MFIVE will autonomously attempt to plan out the entire remaining flight plan. If a situation is reached where some job cannot possibly be scheduled, control will revert to the user. The user can also turn off the AUTO SELECT options at any time by deselecting them from the menu.

Jobs Requiring Multiple Crewmembers

Jobs requiring multiple crewmembers can be scheduled in a manner similar to that for single crewmember jobs. For manual scheduling, after the user has slid into place the rectangle for one of the crewmembers, MFIVE will generate striped rectangles for each of the other crewmembers which are eligible to perform the job at the same start time. The user then clicks on these rectangles until the required number are selected (Figure 42). (If the required number is exactly equal to the number of crewmembers available, then MFIVE will automatically perform the selection.)

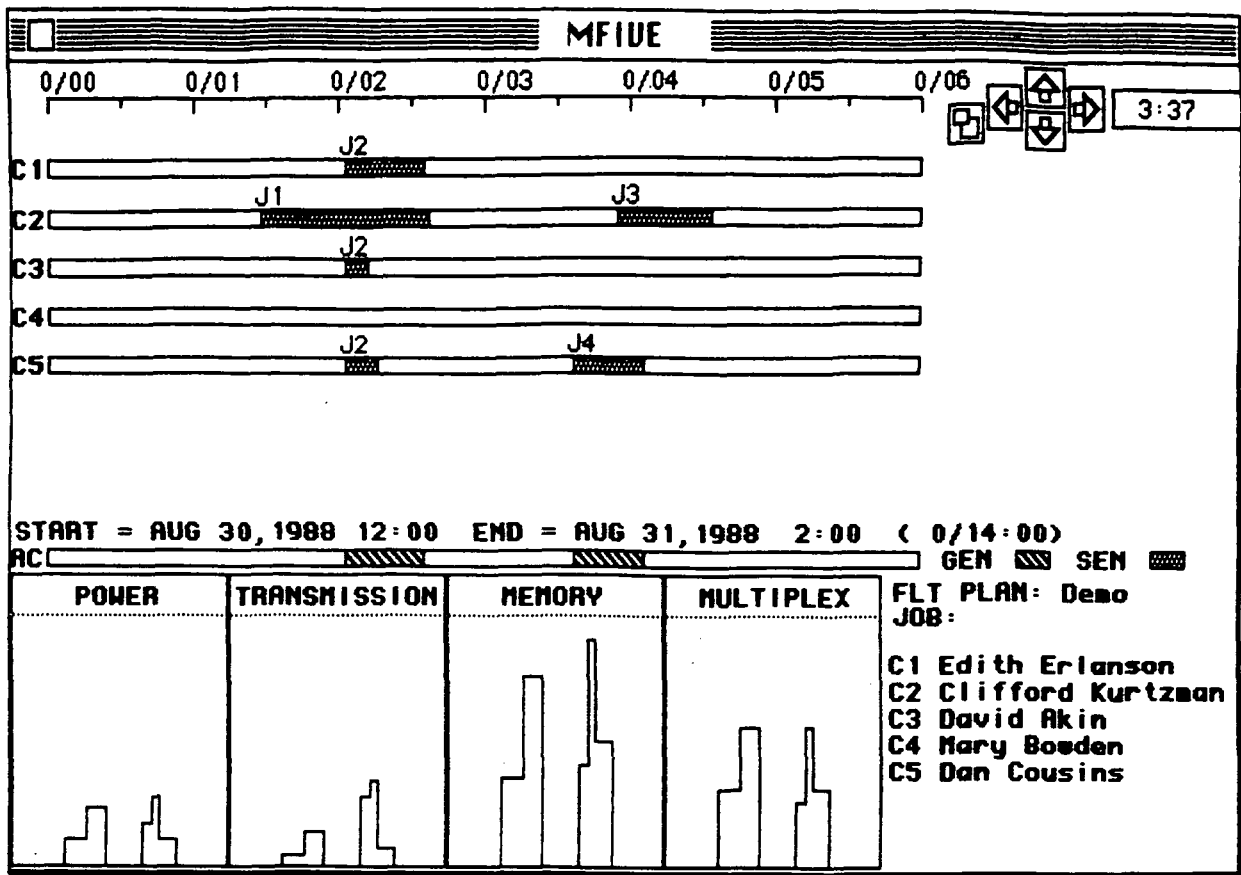


Figure 39

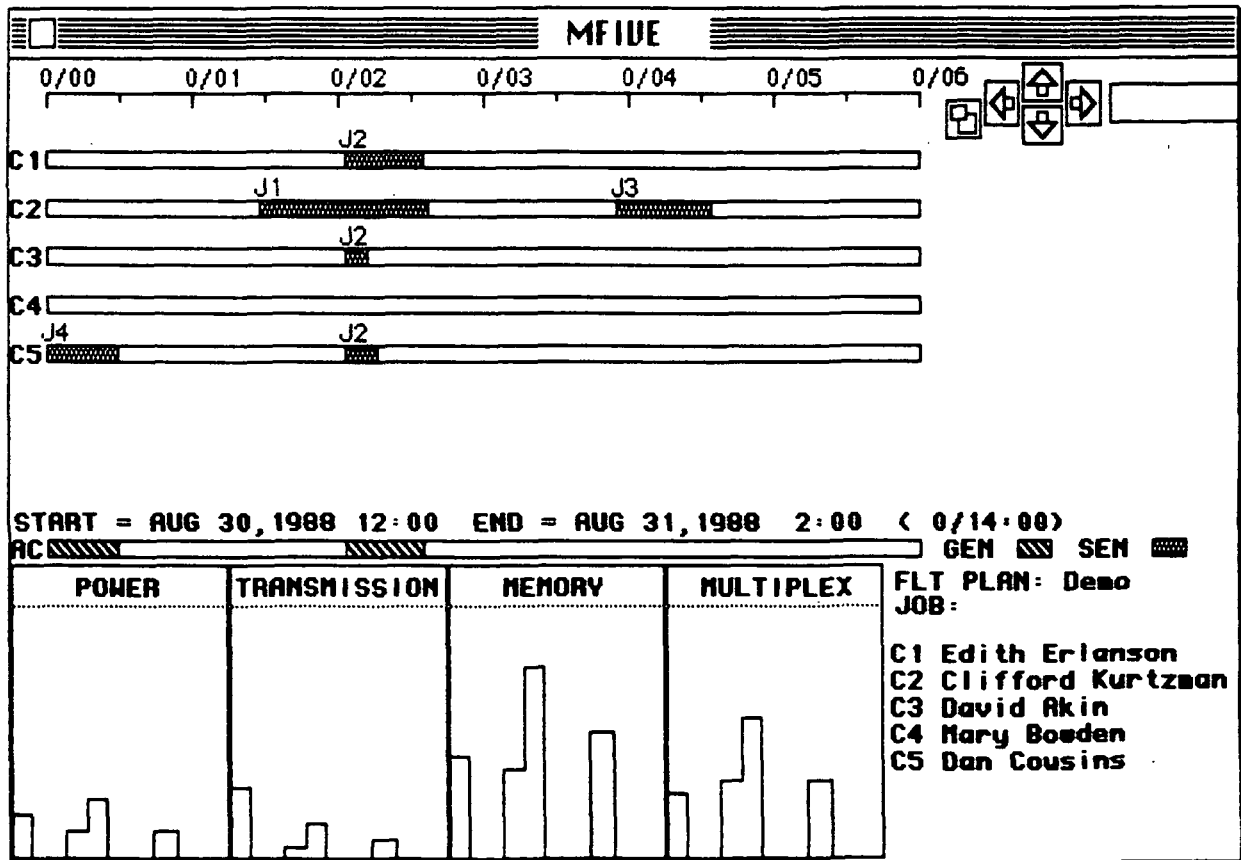


Figure 40

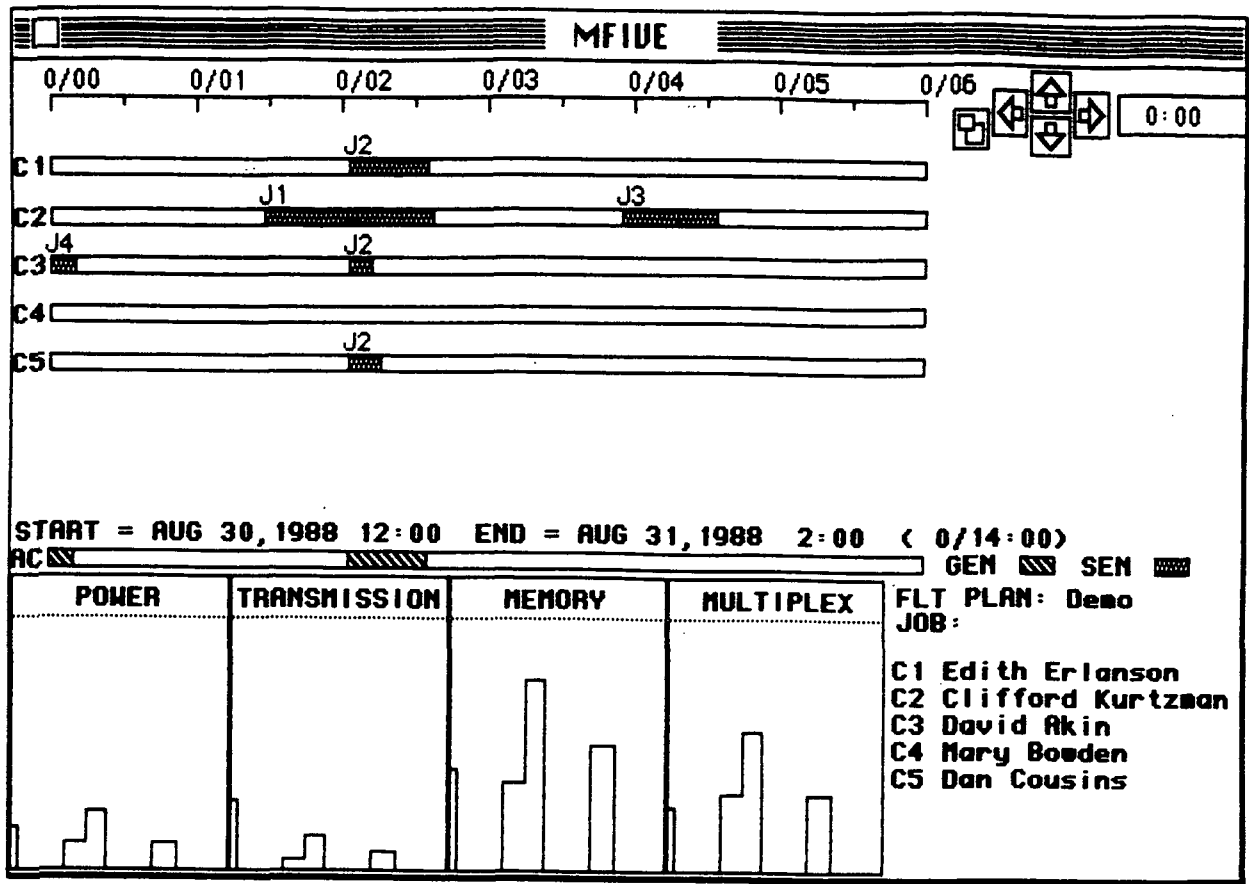


Figure 41

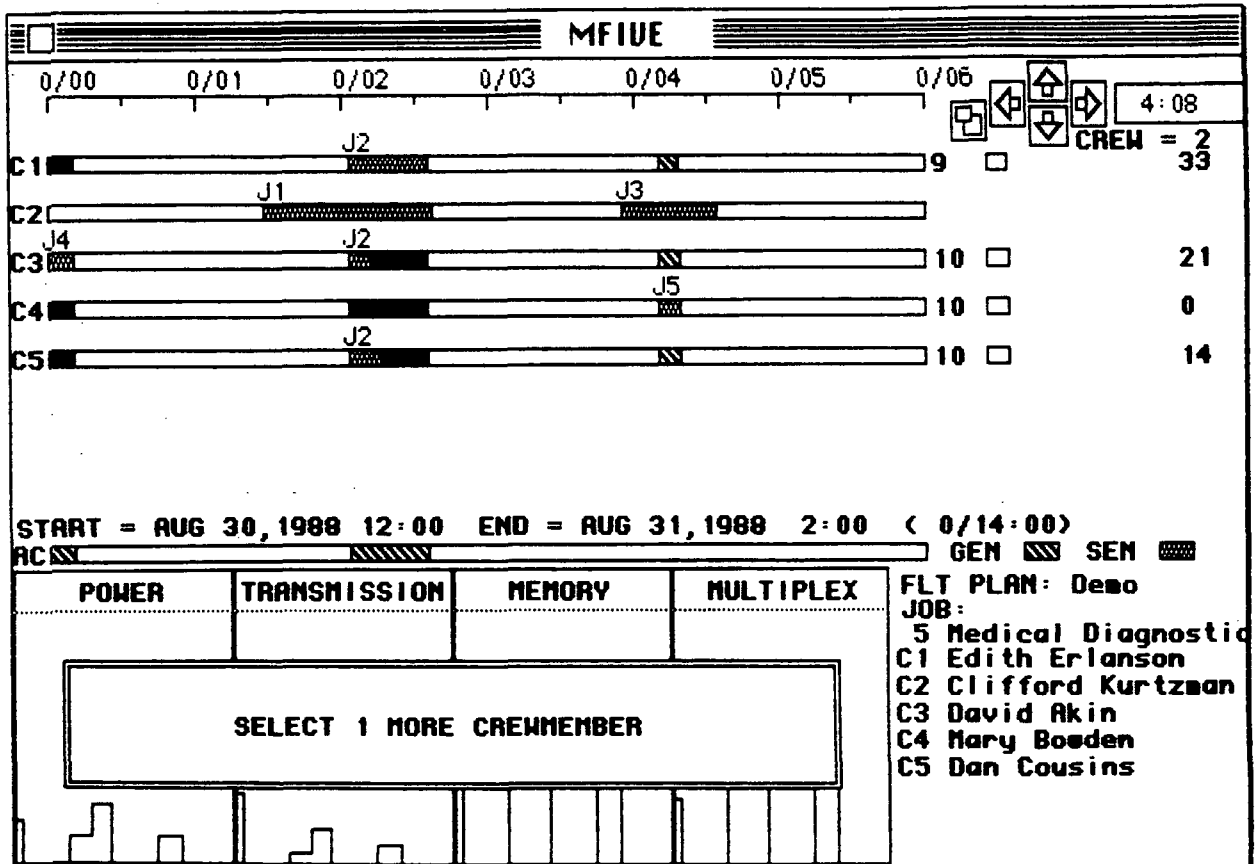


Figure 42

Resource usage is taken to extend over the period covered by the crewmember taking the longest to perform the job.

The job can be assigned to a group of crewmembers at their mutually earliest start time by successively clicking on the numbers of those crewmembers. **SELECT CREW** and **AUTO CREW SELECT** options can also be used to allow MFIVE to autonomously choose crewmembers and time slots for the task.

Time Constraints

The **CONSTRAINTS** menu, Figure 43, allows the user to add time constraints (such as earliest start time and precedence constraints) and target constraints.

Choosing the **TIME CONSTRAINTS** option from the **CONSTRAINTS** menu gives the user a window in which time constraints can be added and modified (Figure 44). Listed in the first four columns of this window are the names of the jobs, their earliest start times (**EST**), latest start times (**LST**), and latest end times (**LET**). For example, in Figure 44 we see that for the job **EXERCISE**, its earliest start time is at the start of the flight plan (i.e., zero days, zero hours and zero minutes). For this same job, the latest start time is at 13 hours and 42 minutes into the flight plan, and the latest end time is at exactly 14 hours into the flight plan (which is the end of the flight plan). In this case, the latest start time is determined because the shortest possible time to perform the job is 18 minutes, and hence it must start no later than 0/13:42 in order to be completed by 0/14:00.

Data can be revised by clicking on an item. For example, in Figure 45, the earliest start time for **MATERIALS PROCESSING** has been selected for revision. Here a new constraint is added, making the earliest start time of this job to be zero days, 3 hours, and 25 minutes into the flight plan. In Figure 46, the user has clicked on the job **EVA** (note also that our previous revision to **MATERIALS PROCESSING** has been entered into the table). By clicking on **EVA**, the last four columns of the table get filled in, expressing time constraints which relate the other jobs to **EVA**. The fifth column, labelled Δ MIN, contains the minimum amount of time separating the start times of each of the jobs from the start time of **EVA**. For example, we see that the job **EXERCISE** could start at most 13 hours and 39 minutes before (as indicated by the negative sign) **EVA**. This would occur if **EXERCISE** were scheduled to start at the beginning of the flight plan and **EVA** were scheduled to start at its latest possible start time, 0/13:39.

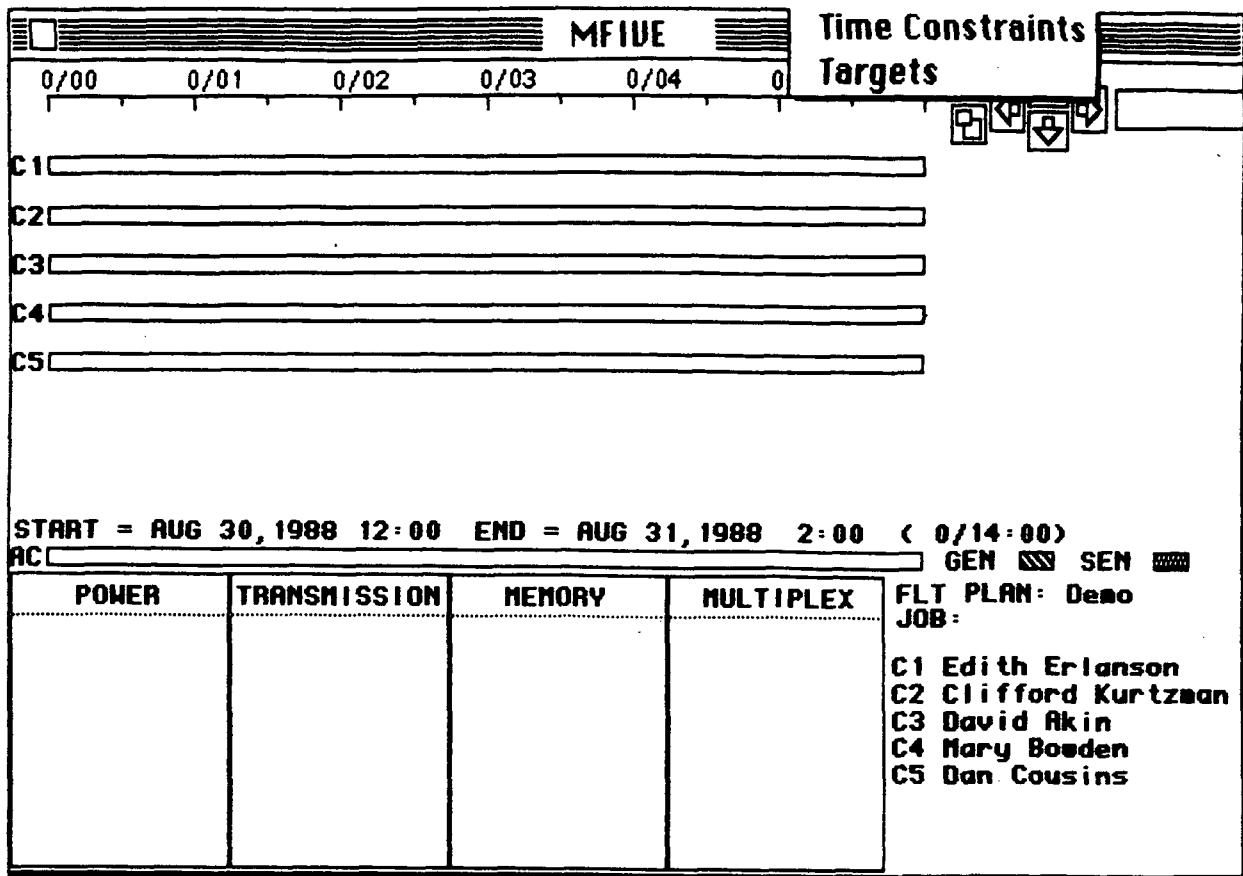


Figure 43

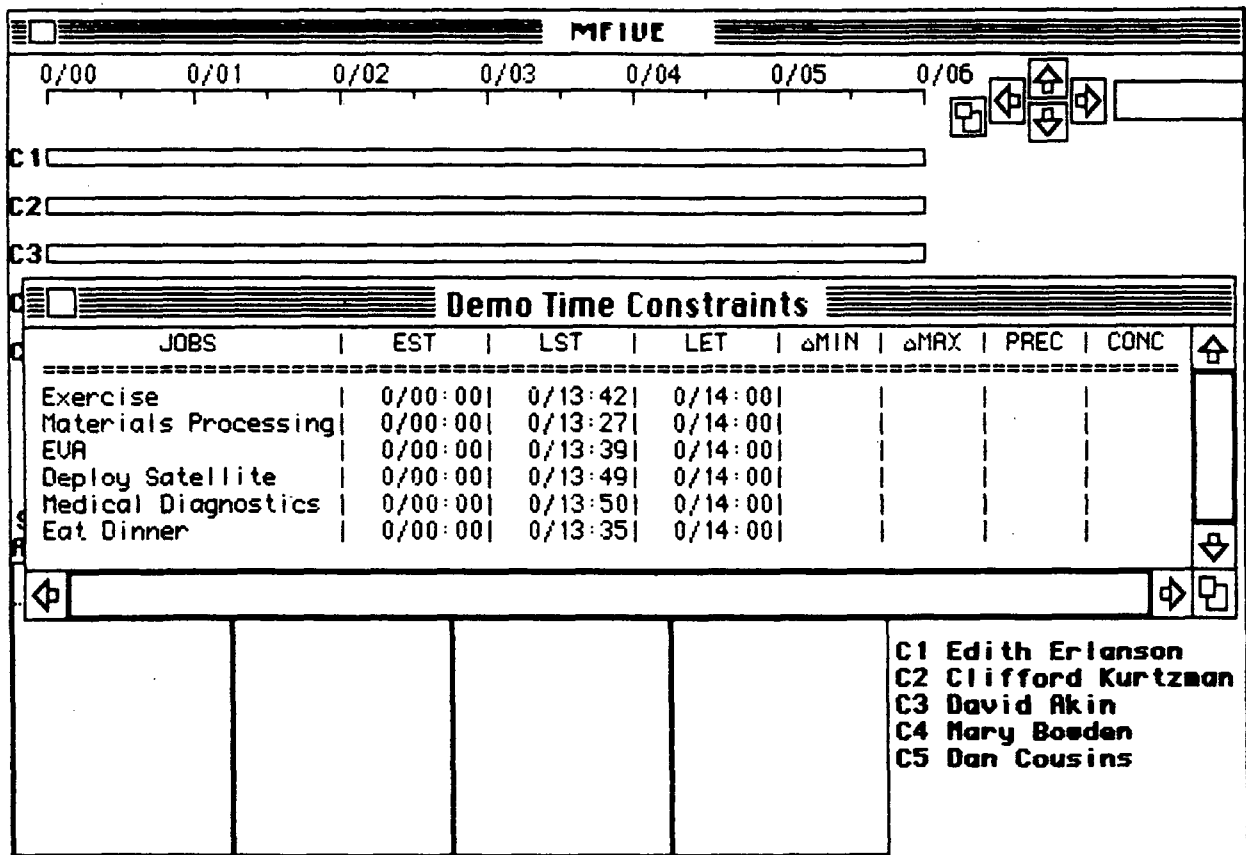


Figure 44

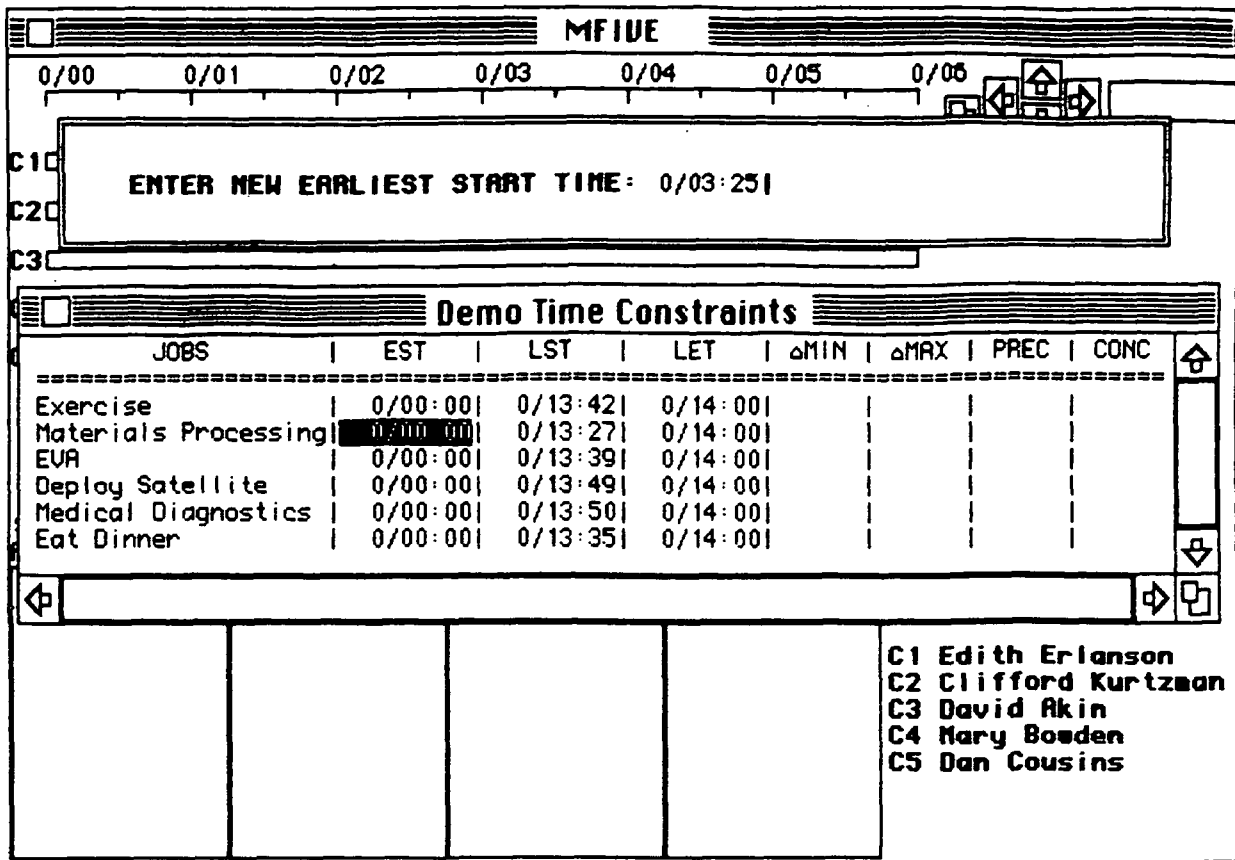


Figure 45

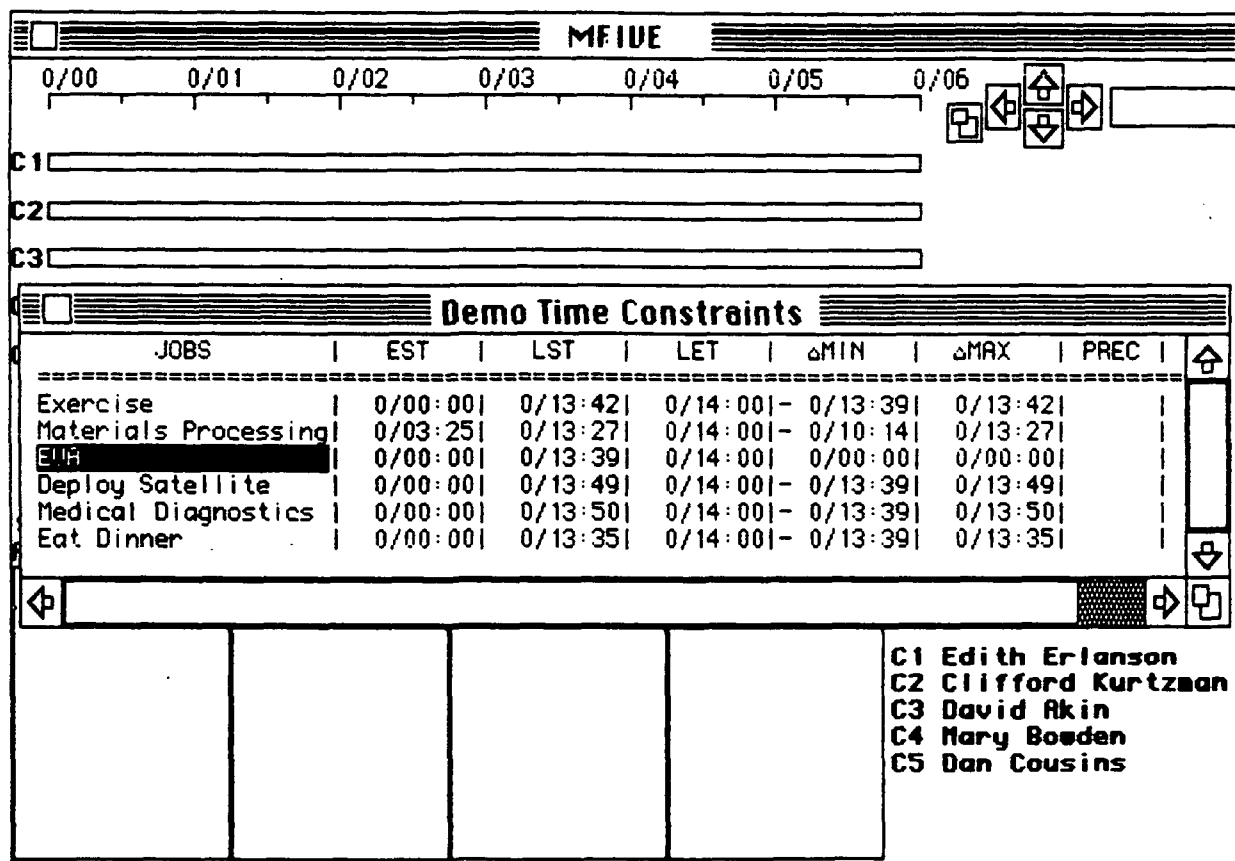


Figure 46

The sixth column, labelled Δ MAX, contains the maximum difference in start time between each of the jobs and EVA. For example, we see that the job EXERCISE can start at most 13 hours and 42 minutes after EVA. This would occur if EVA were scheduled to start at the beginning of the flight plan, and EXERCISE were scheduled to start at its latest possible start time, 0/13:42. Both Δ MIN and Δ MAX constraints can be modified by clicking on the appropriate entry.

The seventh column, labelled PREC, contains precedence constraints between each of the jobs and EVA. Figure 47 shows the process of adding a precedence constraint. The user has clicked on the space for a MATERIALS PROCESSING precedence constraint, and the system has responded with a window asking the user if MATERIALS PROCESSING should occur either before or after EVA. Figure 48 shows the result selecting BEFORE. In addition to including the word BEFORE in the PREC column, it can be seen in Figure 48 that the latest end time of MATERIALS PROCESSING has been changed to 0/13:39, which is the latest start time of EVA. The latest start time for MATERIALS PROCESSING has also been similarly reduced. Also, earliest start time for EVA has been increased to 0/03:58, which is the earliest start time of MATERIALS PROCESSING, 0/03:25, plus the minimum possible job time for MATERIALS PROCESSING, i.e., 33 minutes.

In general, the addition of any new constraint into the constraint table can cause propagation of changes effecting any of the other items in the table. The scheduler utilizes algorithms which make maximum possible inference during constraint propagation. These algorithms are fully discussed in Appendix A of this report.

The eighth column of the constraint table, which is outside the window shown in Figure 48, contains the concurrence constraints. Clicking on the space for a concurrence constraint allows the user to require the start times of any two jobs to be the same.

The system will not permit the user to enter mutually contradictory constraints. For example, after telling the system that MATERIALS PROCESSING is before EVA, one could not enter a concurrence constraint between those two jobs. It can also be seen that earliest start time and Δ MIN values must always increase, while latest start time, latest end time and Δ MAX values must always decrease, if constraint contradiction is to be avoided.

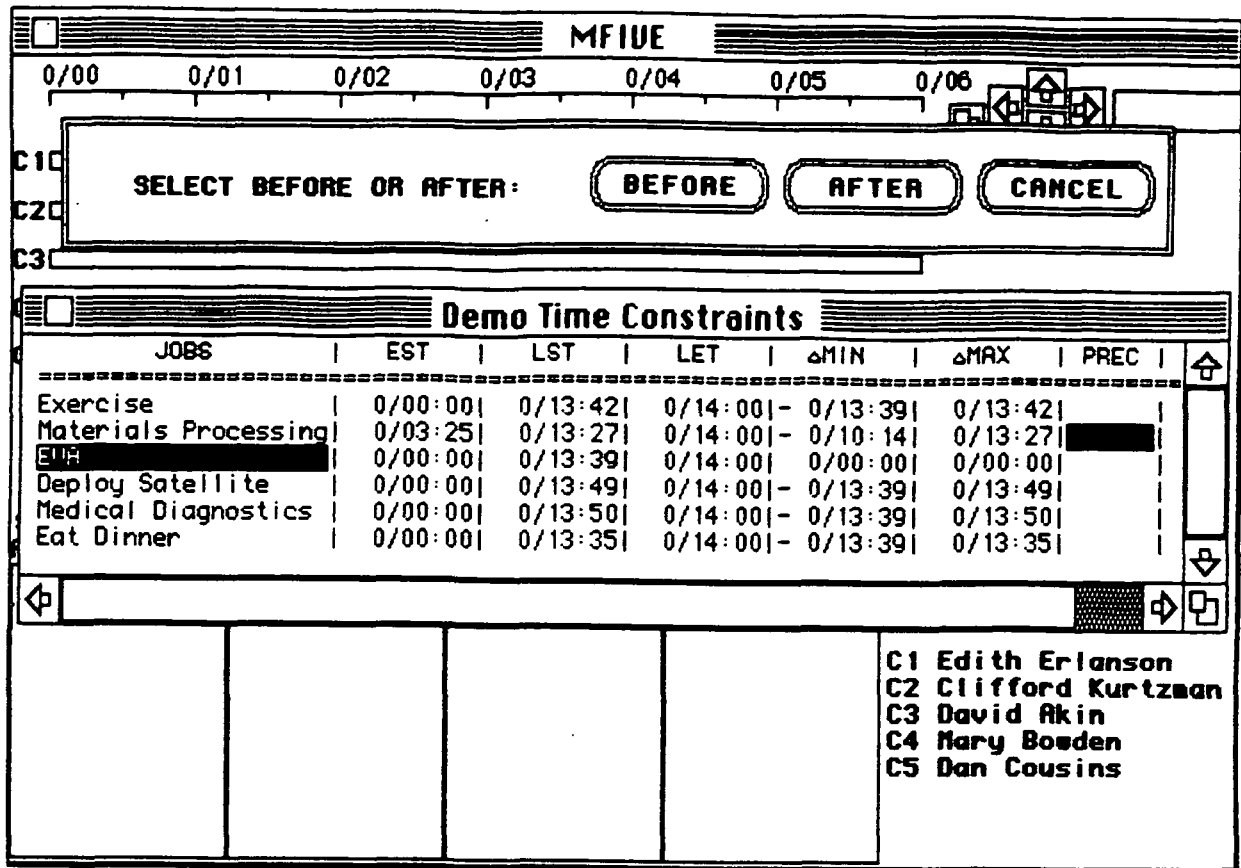


Figure 47

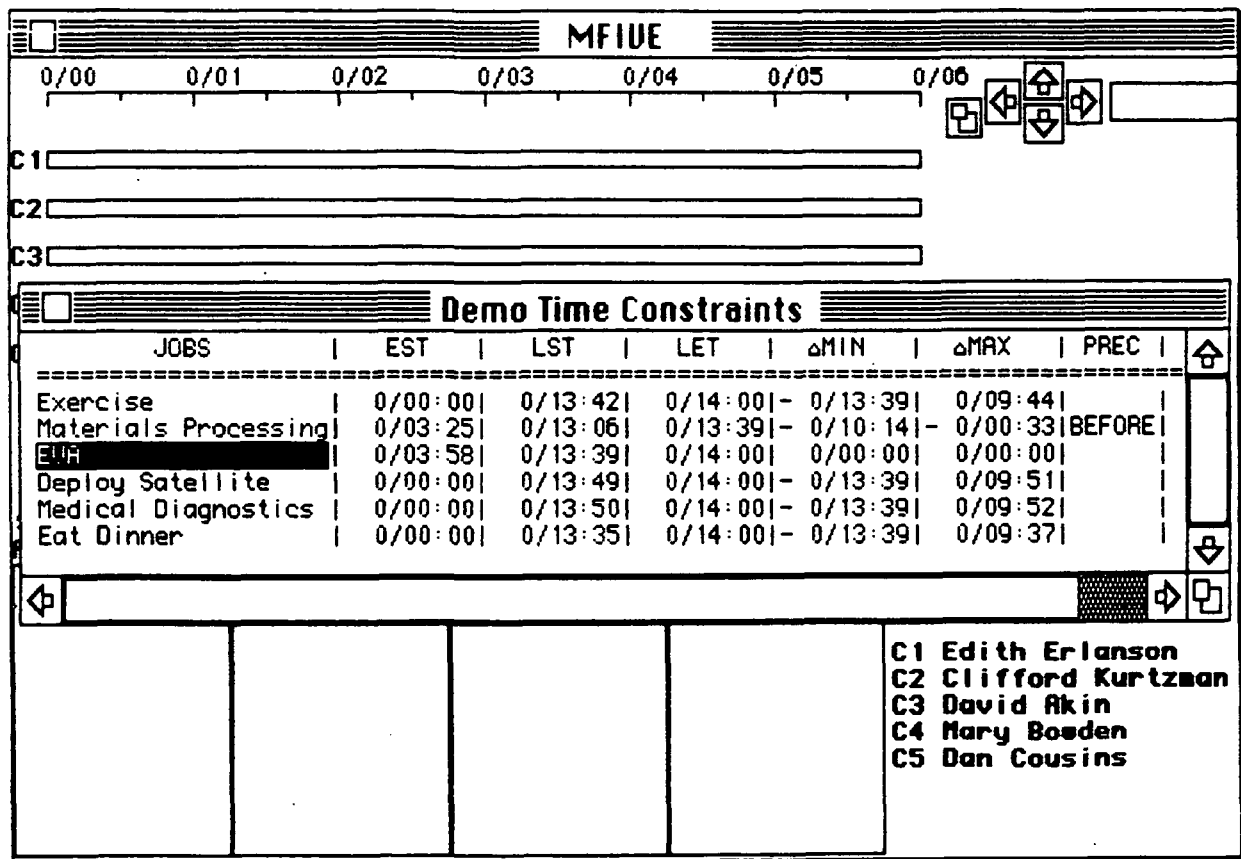


Figure 48

At present, there is no way for a user to delete a previously added constraint, other than to restart the scheduling of the flight plan. It is planned that future versions of the scheduler will have this option, but providing it is difficult because removing a constraint would in addition involve removing all of the inferences made from that constraint.

Target Constraints

In actual operation, target constraints (i.e., time windows during which a task can or cannot be performed) can be specified by the satisfaction of a large number of varied functions, such as meeting a desired set of orbital parameters. While explicitly modeling each possible source of target constraints is an interesting problem which will have to be addressed in a fully operational system, it is not crucial to the scheduler's operation. From the point of view of the scheduler, it is only the final time windows which are relevant, and not the functions which produce them.

Selecting the TARGETS option from the CONSTRAINTS menu allows the user to specify windows during which performance of a job is not feasible. The user is first given a listing of jobs to select from (Figure 49). After selecting a job, a box appears (Figure 50) allowing the user to add, modify, or delete windows during which the job cannot be performed. Figure 50 shows, for the job EXERCISE, its earliest start time, latest start time, latest end time, and two windows during which it has been specified as infeasible. Clicking on the number of a window will allow the user to delete it. Window start or end times can be revised by clicking on them. Scrolling the information to the bottom will allow the user to add additional windows. After closing this box (by clicking on "done") and selecting EXERCISE for scheduling (Figure 51), it can be seen that the infeasible windows have been blacked out, and the job therefore cannot be scheduled during these periods.

Other Options

The SAVE SCHEDULE, LOAD SCHEDULE, and DELETE SCHEDULE entries on the OPTIONS menu (Figure 35) allow the user to save and reload an entire schedule at any point in the planning process. This also enables the user to compare and contrast several different schedules prepared from the same initial database.

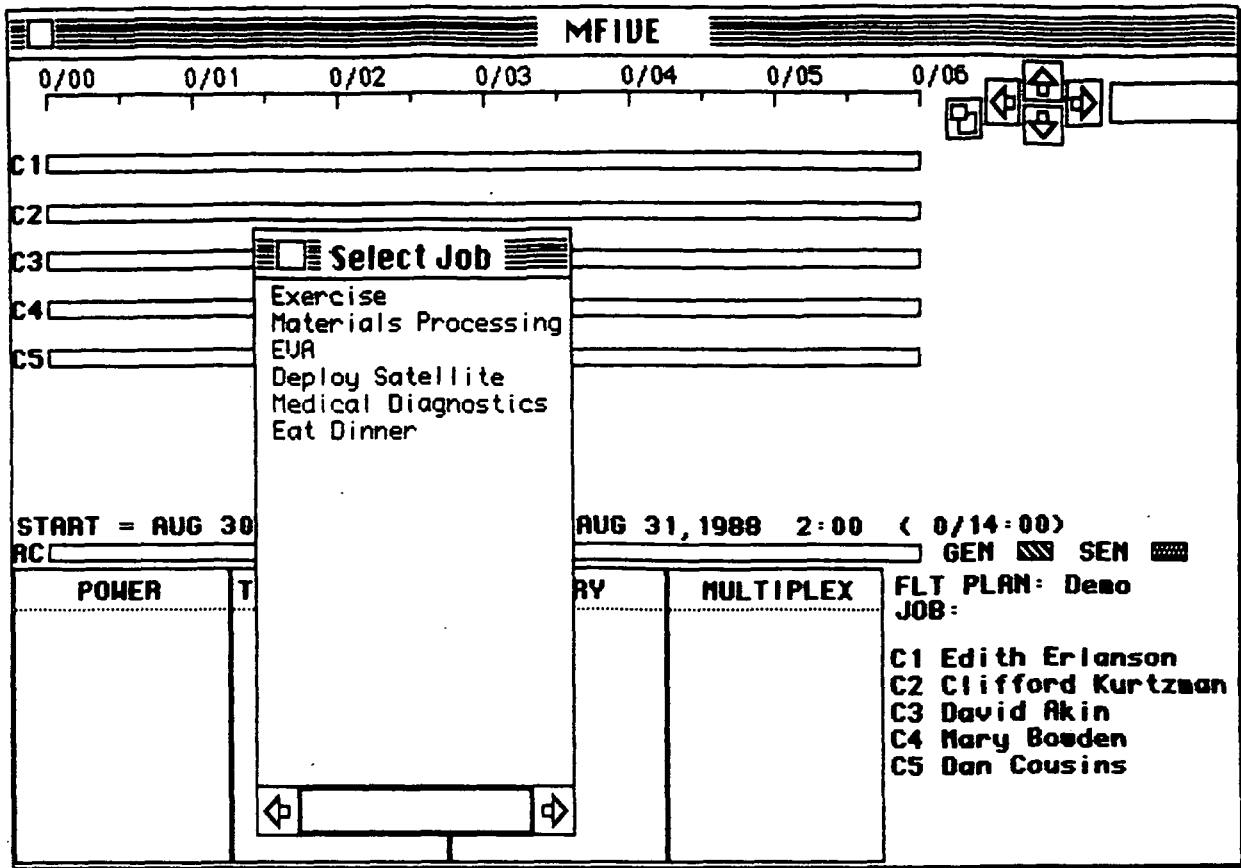


Figure 49

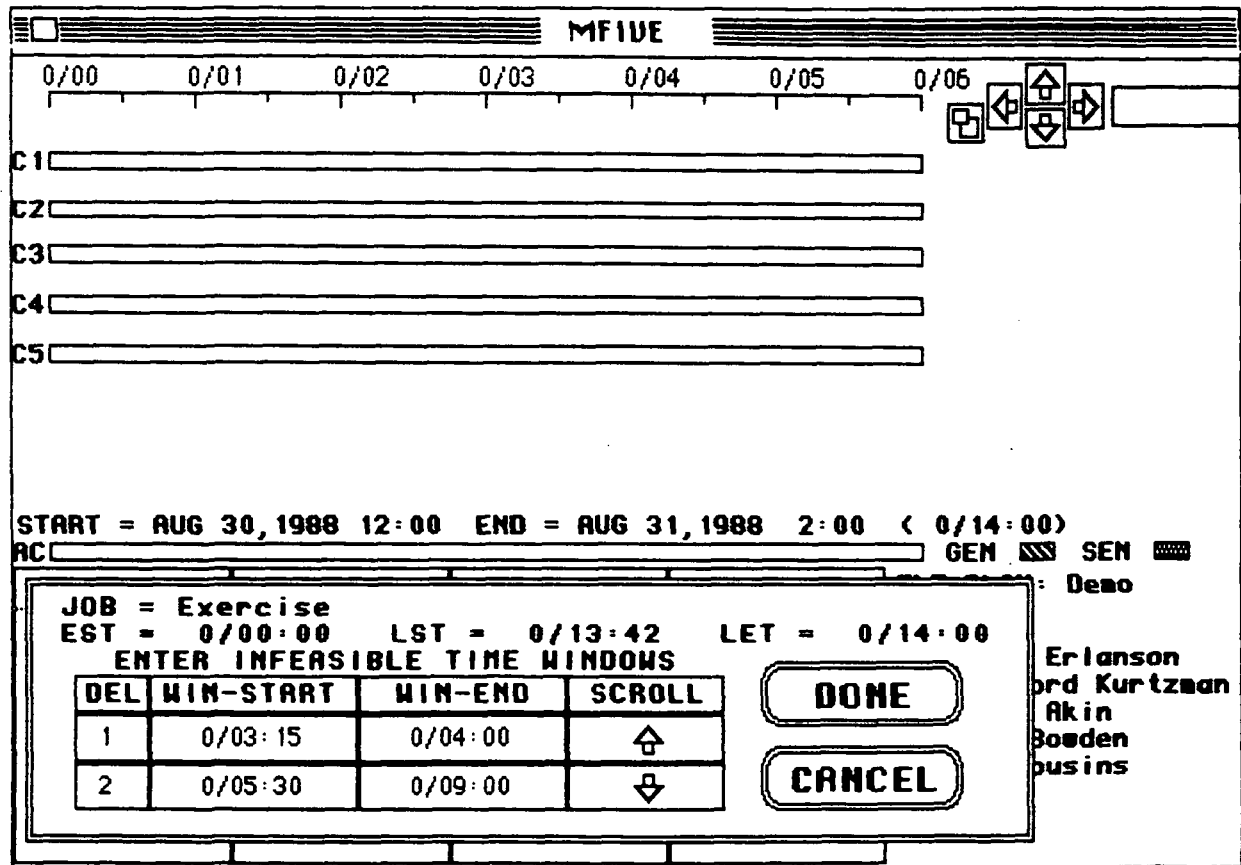


Figure 50

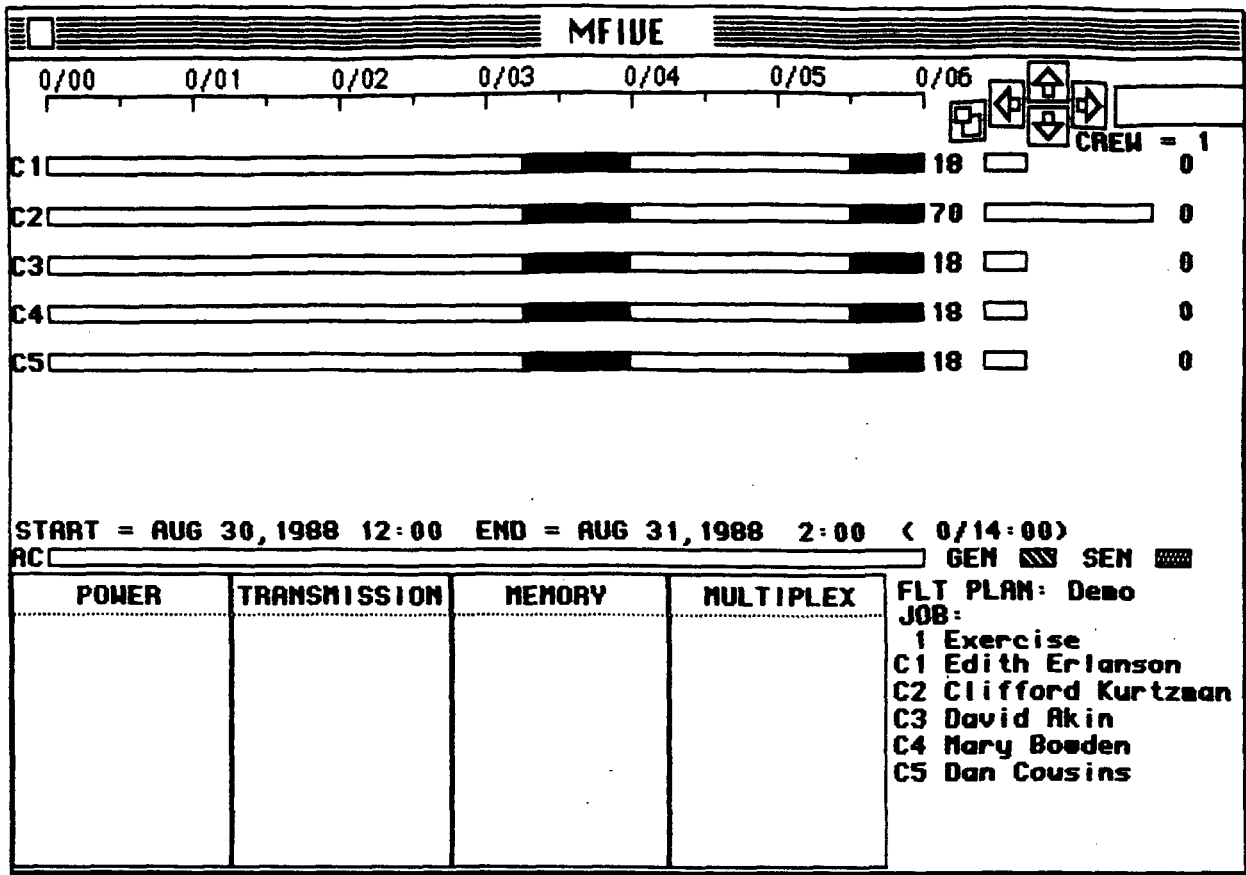


Figure 51

The PRINT SCHEDULE option allows the user to print out a schedule (Figure 52). Schedules can be depicted by job, crewmember, and by pictorial representation of the timeline.

4.2.3 The Tool Searcher

Clicking the mouse on the SEARCH button on the tool worksheet (Figure 21) initiates the tool searcher. The tool searcher (Figures 53 and 54) presents the user with the default location of each copy of the tool. The default location is the place in which the system has been told the tool should be kept. The tool searcher also gives a ranking of other possible locations in which the tool might be found if it is not in its default (nominal) location. The ranking of a stowage compartment is based on factors such as the volume of the compartment, its distance from the default location, its distance from the place of last use, and whether the tool has been found there before (see Section 3.5). Finally, the tool searcher presents the user with a list of the crewmembers who have recently used the tool and the compartments in which it was used.

The tool searcher utilizes a goal-directed approach (backward chaining) when generating a ranking of stowage compartments. It begins the search with a list of all of the compartments onboard the station. The searcher then steps through the production rules by assigning point values to each compartment, based upon each production rule's evaluation. Some compartments will be eliminated from the list (e.g., for being too small to contain the missing item). The production rules process each of the compartments via criteria which are independent of all of the other compartments; the complexity of the ranking algorithm therefore increases linearly with the number of bins. Finally, the results of the ranking algorithm are sorted for presentation to the crewmembers. The complexity of a sort increases as $(n \times \ln n)$, where n is the number of compartments, and $\ln n$ is the natural logarithm of the number of compartments.

The tool searcher asks the user whether or not the tool was indeed found. If it was not found, the status of all the copies of the tool are changed to LOCATION UNKNOWN. If it was found, the user is presented with a window (Figure 55) and asked to select the compartment(s) in which the tool was located. This information is then used to increase evidence for finding the tool in these compartments the next time a search is conducted.

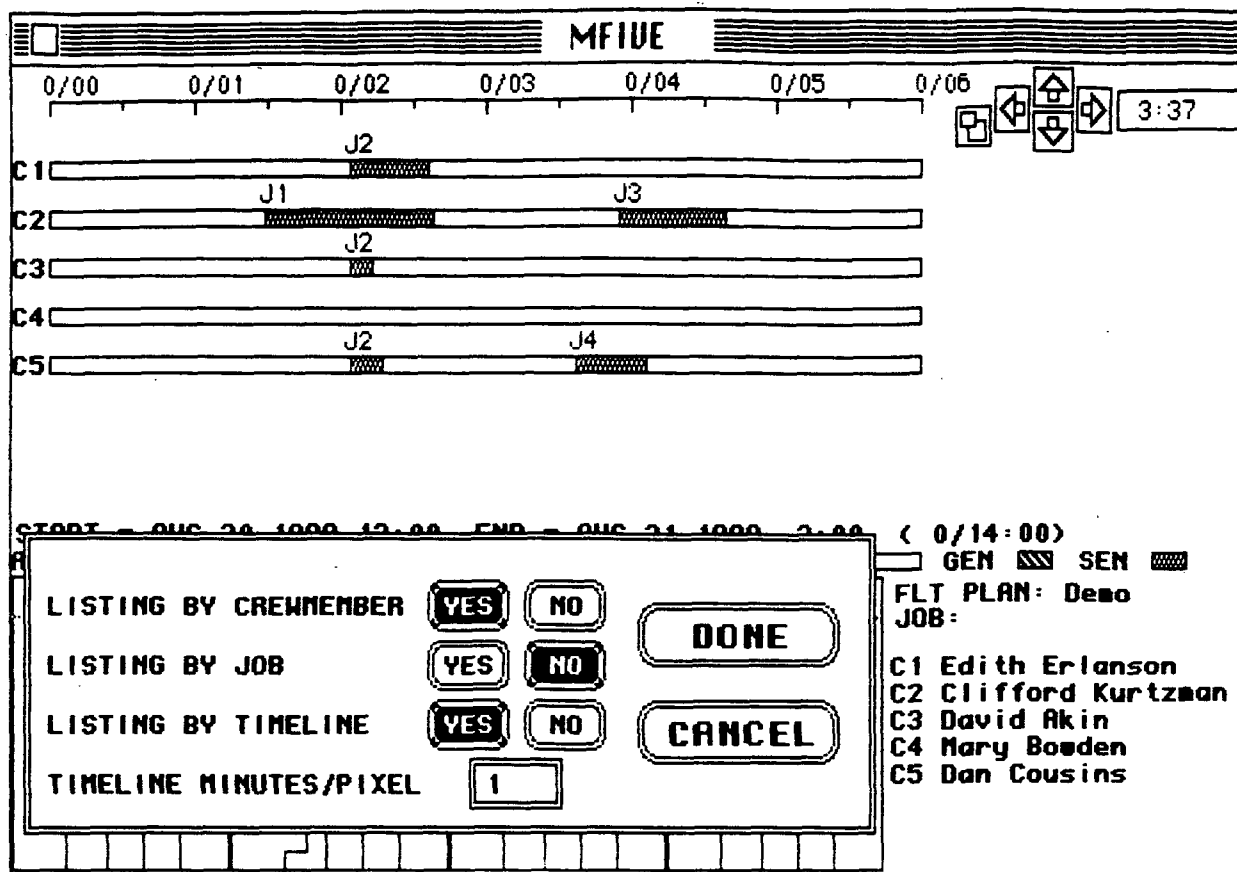


Figure 52

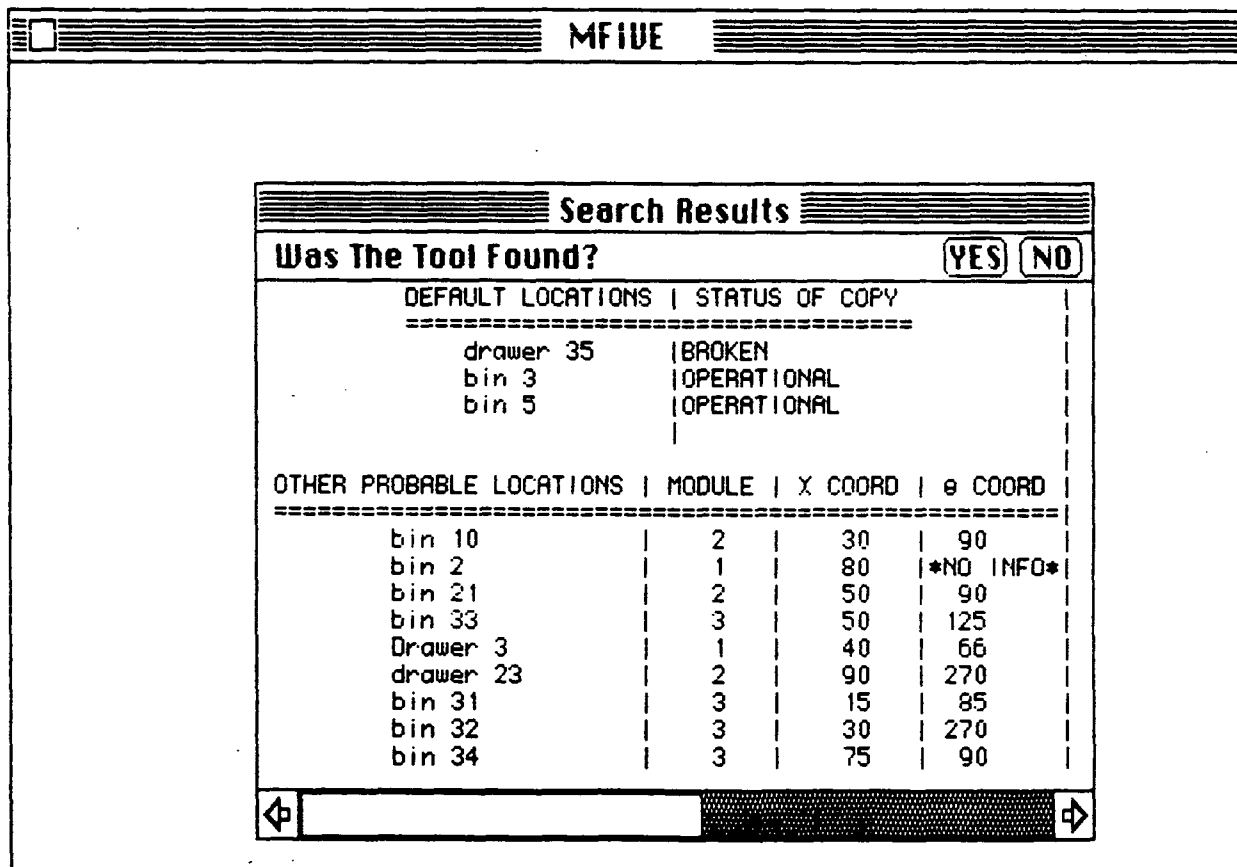


Figure 53

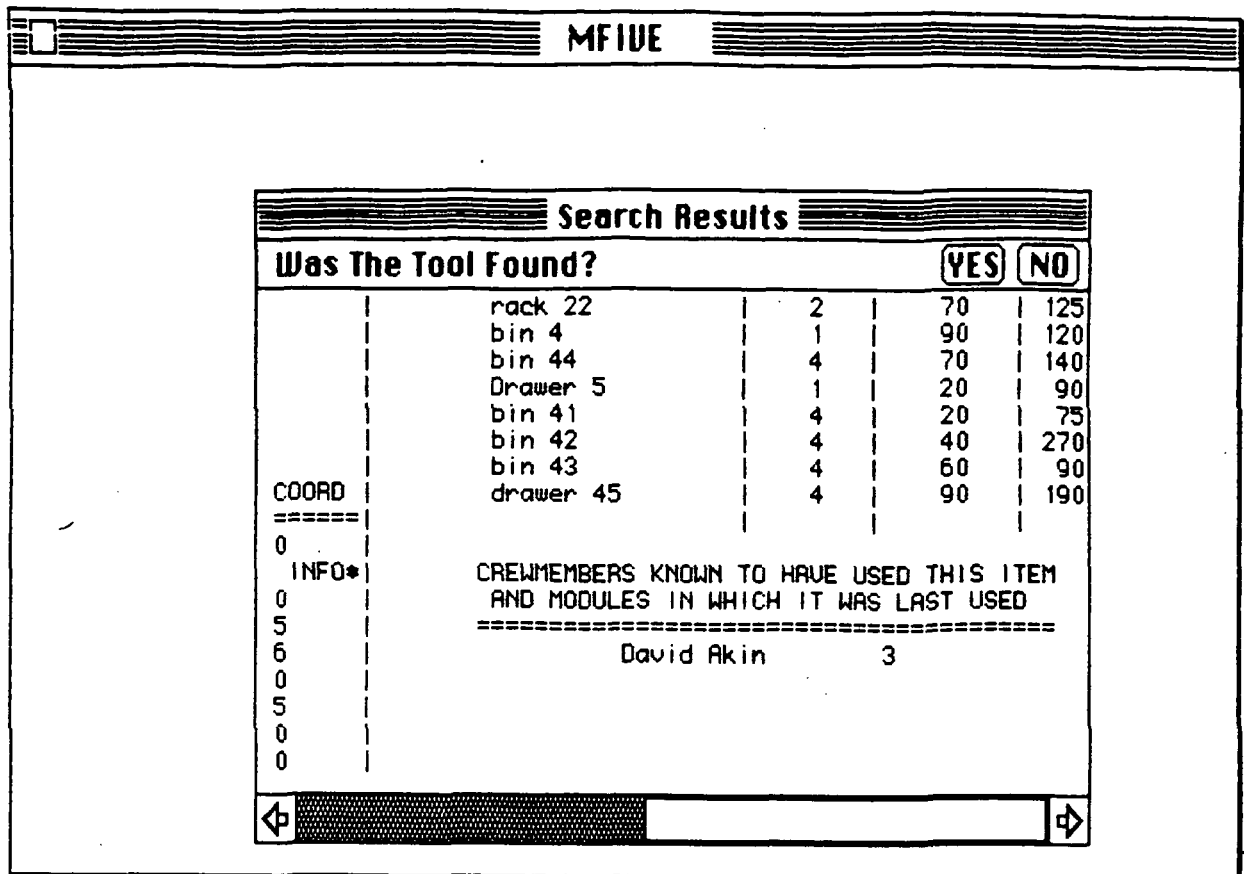


Figure 54

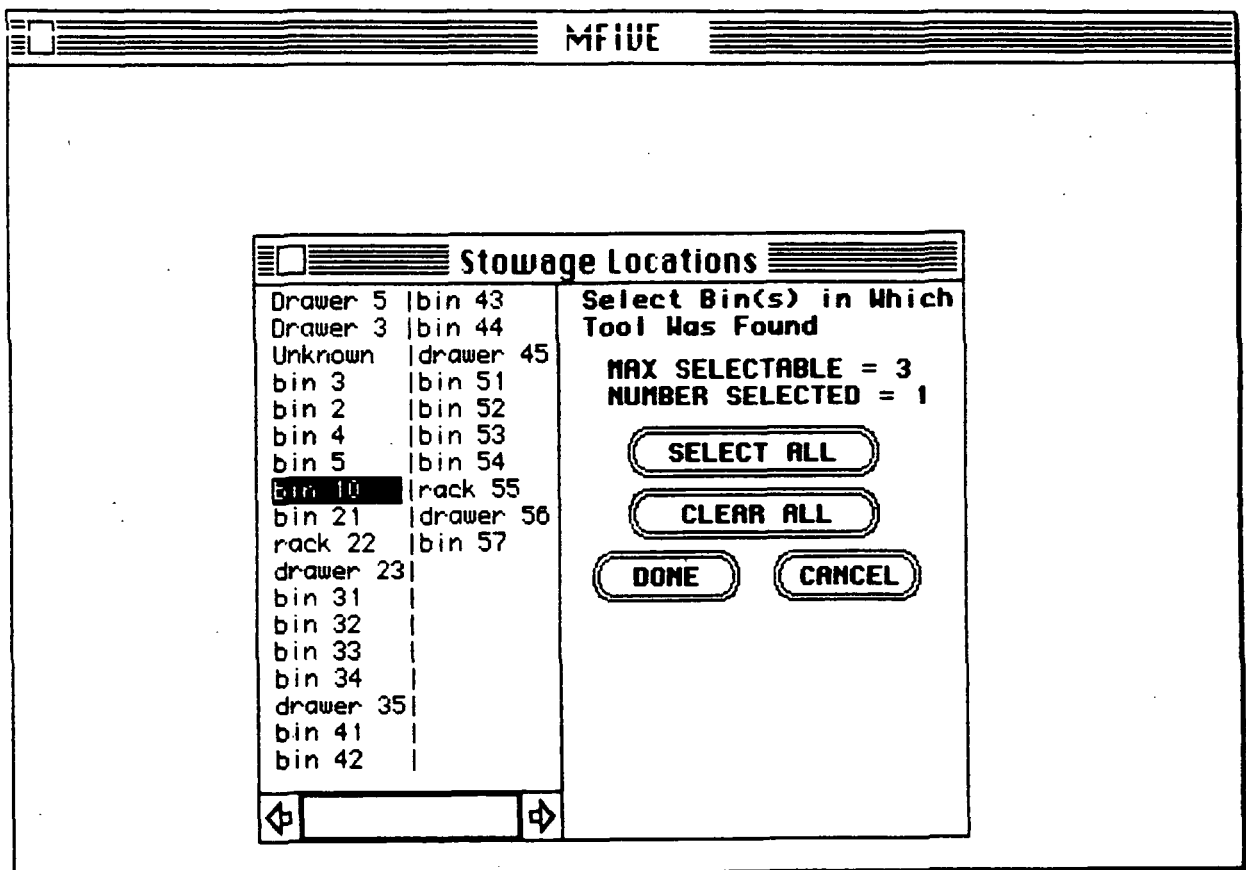


Figure 55

Section 5: Conclusions and Recommendations

The objectives of this study were to evaluate the techniques available pertaining to space station crew activity scheduling, to develop a prototype system to demonstrate the potential of computer based schedulers, to test scheduling algorithms, and to develop a computer based stowage logistics clerk. On the whole, the study was successful. As this report has described, an extensive literature search was conducted, and a demonstration scheduling/logistics clerk system has been developed. This system can perform the testing of various algorithms for crew activity planning.

The following, however, are areas in which difficulty was experienced, or in which initial expectations were not fully met:

- 1) Choosing the proper computer system to use to develop the software was a significant problem. Ideally, a Lisp machine would have been used, but none was available to this study and the cost of procuring one was excessive. Several exploratory efforts were made as the development of the scheduling system was switched from a mainframe computer to an IBM personal computer, and finally to an Apple Macintosh. After a delay of many months, however, the results available on the Macintosh proved to justify the additional effort. The resulting hardware system is inexpensive, while illustrating user friendly interfaces which make the scheduler very simple to operate.
- 2) A major disappointment was the inability to interact directly with scheduling personnel at NASA, particularly at the Johnson Space Center (JSC). There were several factors contributing to this problem. Several unsuccessful attempts were made to contact JSC scheduling personnel. At one point, the study team asked a former colleague, who now works at JSC, to attempt to acquire information and make contacts: he received a reprimand for his efforts. It is hoped that future efforts to make appropriate contacts will be more successful.
- 3) The scheduler, as presented in this report, currently lacks several features which will be necessary in a fully operational system. These features include:

- A) The ability to schedule and display tasks which do not require any crewmembers to perform. For example, a material processing experiment might consist of sample preparation, sample processing, and analysis. The processing step, such as baking the sample in an oven for many hours, might not require any crewmembers. It is important, however, to include this step in the plan timeline.
- B) The ability to schedule groups of jobs as a unit. For example, one would prefer to schedule the above materials processing experiment as a single unit, rather than as three separate steps related by precedence constraints.
- C) The ability to schedule multiple repetitions of a job. For example, one would like to specify that sleep periods that begin every 24 hours, rather than specifying each sleep period as a separate task.
- D) The interface between the scheduler and the stowage clerk is currently simulated. The scheduler knows which tools are necessary to perform each job, and will eventually be able to tell the logistics clerk who last used a given tool and where it was last used.

The study has arrived at the following recommendations for future research:

- 1) Continued development of the MFIVE system should implement the features described in item 3, above.
- 2) Standard scheduling problems should be established to use as a baseline for the testing of scheduling algorithms. Ideally, these would consist of actual Space Shuttle/Spacelab mission planning problems in their full complexity. Cooperation is needed from JSC scheduling personnel in obtaining a rigorously defined and detailed scenario.
- 3) The system is now capable of performing testing and tradeoff analysis of various types of scheduling algorithms, and continued research should be done

to determine the best optimization techniques for forming a good schedule in a problem of this complexity.

- 4) A more advanced logistics clerk expert system, capable of learning from its own mistakes, should be designed to provide good guesses as to where a missing item might be located.
- 5) The use of an extensive voice-recognition system, which could be installed in each module within easy access of the space station's stowage compartments, should be investigated [Riley and Vestewig, 1983]. With such a system, the crewmembers would be able to easily inform a computer based logistics clerk every time they remove an item from its container and every time they put it back. The logistics clerk would then have much more certain knowledge as to a missing item's location.

The basic computer techniques by which these recommendations can be implemented either already exist, or will exist in the near future. The ultimate objective of these recommendations is to provide the crewmembers with a means of solving problems by themselves, without having to rely on ground control.

Appendix A: An Optimal Algorithm for the Propagation of Time Constraints

A.1 Problem Overview

In this appendix a methodology is developed for propagating generalized time constraints relating tasks to other tasks and fixed points in time. It is shown that the constraints can be transformed into a format which will allow, for each task, maximum inference of feasible time windows via an application of the shortest path algorithm. Efficient methods for implementing the algorithm are then discussed.

Consider the problem of scheduling a set of n tasks. Numerous types of time constraints may be specified for each task. Some constraints relate performances to fixed points in time. Earliest start time constraints, $EST(x)=\mu$, specify that task x must begin no earlier than time μ . Latest start time constraints, $LST(x)=\mu$, specify that task x must begin no later than time μ . Required start time constraints, $RST(x)=\mu$, specify that task x must start at time μ . Finally, latest end time constraints, $LET(x)=\mu$, specify that performance x must be completed by time μ . It should also be noted that every schedule has some minimum time, MS , before which no task may be scheduled and some maximum time, ME , after which no task may be completed.

Other constraints relate tasks to other tasks. Precedence constraints, $P(x,y)=1$, specify that task x must be completed before the start of task y . Concurrence constraints, $CO(x,y)=1$, specify that tasks x and y must begin at the same time. Minimum delta start time constraints, $\Delta MIN(x,y)=\mu$, indicate that there must be at least an interval of length μ between the start of task x and the start of task y . Maximum delta start time constraints, $\Delta MAX(x,y)=\mu$, indicate that the maximum interval between the start of task x and the start of task y is μ .

For each task k ($k=1, 2, 3, \dots, n$), we can compute the values $MAXT(k)$ and $MINT(k)$ where $MAXT(k)$ is the maximum time that it could take to perform task k and $MINT(k)$ is the minimum time that it could take to perform task k . For example, if k is a task requiring one crewmember, then $MINT(k)$ is just the performance time of the crewmember who can complete the task fastest. In the absence of more complete information, one can always set $MINT(k)=0$ and $MAXT(k)=\text{infinity}$ (or some other large value, such as $ME-MS$).

Propagating the constraints can yield numerous inferences about the feasible time windows for each task. For example, if $P(x,y)=1$, then the earliest start time for task y can be inferred to be at least as late as the earliest start time for task x plus the minimum time of task x , i.e., $EST(y) \geq EST(x) + MINT(x)$. What is desired is an algorithm for making maximum inference from the specified constraints to narrow down the feasible time windows for each task.

A.2 Constraint Reduction

The first step in reducing the problem is to add an event $*$ which has zero duration ($MINT(*)=MAXT(*)=0$) and required start time MS ($RST(*)=MS$).

The second step is to change all constraints to ΔMIN and ΔMAX constraints. This is accomplished by the following transformations:

$$\begin{aligned} EST(x) = \mu & \text{ becomes } \Delta MIN(*,x) = \mu - MS \\ LST(x) = \mu & \text{ becomes } \Delta MAX(*,x) = \mu - MS \\ RST(x) = \mu & \text{ becomes } \Delta MIN(*,x) = \Delta MAX(*,x) = \mu - MS \\ LET(x) = \mu & \text{ becomes } \Delta MAX(*,x) = (\mu - MS) - MINT(x) \\ P(x,y) = 1 & \text{ becomes } \Delta MIN(x,y) = MINT(x) \\ CO(x,y) = 1 & \text{ becomes } \Delta MIN(x,y) = \Delta MAX(x,y) = 0 \end{aligned}$$

(It should be noted that $\Delta MAX(*,x) = (\mu - MS) - MINT(x)$ is necessary but not sufficient for $LET(x) = \mu$, and that $\Delta MIN(x,y) = MINT(x)$ is necessary but not sufficient for $P(x,y) = 1$. However, until event x is scheduled, we cannot make any better inference from these constraints than is given above.)

The third step is to change all ΔMIN constraints to ΔMAX constraints. This is done by replacing all $\Delta MIN(x,y) = \mu$ by $\Delta MAX(y,x) = -\mu$

Finally, the last (fourth) step is to form the complete ΔMAX matrix. This is accomplished by initializing:

$$\begin{aligned} \Delta MAX(k,*) &= 0 & k &= *, 1, 2, \dots, n \\ \Delta MAX(k,k) &= 0 & k &= 1, 2, \dots, n \end{aligned}$$

$$\Delta\text{MAX}(m,n) = (\text{ME} - \text{MS}) - \text{MINT}(n) \quad \begin{array}{l} m = *, 1, 2, \dots, n \\ n = 1, 2, \dots, n \\ m \neq n \end{array}$$

and then substituting each of the ΔMAX values found in steps 1 - 3 into the ΔMAX matrix whenever they are less than the corresponding value already entered in the matrix.

A.3 Inference Generation

Time windows for each of the tasks may now be found by applying a shortest path algorithm (such as Floyd's or Dijkstra's) to the ΔMAX data by considering each of the tasks as a node and each of the constraints as an arc on a graph. For example, if $\Delta\text{MAX}(*,1)=5$, $\Delta\text{MAX}(1,2)=8$, and $\Delta\text{MAX}(*,2)=20$, then we can replace $\Delta\text{MAX}(*,2)$ by 13 because the first two constraints form a tighter limit (shorter path) on the maximum interval between the start of the timeline and the start of task 2.

Let the shortest path algorithm yield the matrix ΔMAXS , containing the length of the shortest path from every node to every other node in the ΔMAX graph. Then the latest start time (as inferred) for each task is then given by $\text{LST}(x) = \text{MS} + \Delta\text{MAXS}(*,x)$, $x=1, 2, \dots, n$.

Similarly, we can find the inferred earliest start times for each of the tasks: $\text{EST}(x) = \text{MS} - \Delta\text{MAXS}(x,*)$, $x=1, 2, \dots, n$. For example, if $\text{MAXS}(x,*) = -8$, then there is at least 8 time units between the start of the schedule and the start of task x .

If time windows become so constrained that the latest start time for a task is before the earliest start time ($\text{LST}(x) < \text{EST}(x)$ or alternately $\Delta\text{MAXS}(x,x) < 0$) then we can infer that there is no feasible time at which task x can be scheduled for which it satisfies all of its time constraints (in other words, there has been some logical contradiction implied in specifying the time constraints).

A.4 Algorithms

Shortest path problems can generally be solved using algorithms such as Floyd's or Dijkstra's (revised for graphs with negative cost arcs) [Larson and Odoni, 1981]. There are several special properties of this constraint problem which can be exploited to yield more efficient implementation.

Floyd's algorithm has the advantage of being very easy to program (it requires only two short lines of code in APL). Floyd's algorithm involves on the order of n^3 computation, where n is the number on nodes in a graph. It produces the shortest distance from all nodes to all other nodes. Further, there is the added advantage that as new arcs are added to a network, (i.e., by adding a new constraint), the table of shortest paths can be updated with only on the order of n^2 computations. This can be shown by examining Floyd's algorithm. Let D_i be the table of shortest paths at iteration i of the algorithm, and let D_0 be the initial Δ MAX matrix produced in step 4, as described above. Then Floyd's algorithm is:

STEP 1: Set $i = 1$

STEP 2: Set $D_i = \text{Min} \{D_{i-1}, B_i\}$

where the minimization is across corresponding elements of D_{i-1} and B_i and where B_i is defined as the generalized outer product with respect to addition of the i th column of D_{i-1} and the i th row of D_{i-1} .

$$\text{i.e., } B_i(j,k) = D_{i-1}(j,i) + D_{i-1}(i,k)$$

STEP 3: If $i=n$ stop; if $i < n$ then set $i = i + 1$ and go to Step 2.

Step 2 of the algorithm involves n^2 computations and it is repeated through n iterations; hence the algorithm is of the order of n^3 . To incorporate (and thereby propagate) a new arc $J(x,y)=\mu$ into a reduced shortest path matrix J (i.e., one which is the result of Floyd's algorithm), where $\mu < J(x,y)$, all one has to do is run Step 2 of the algorithm twice on J : once with $i=x$ and once with $i=y$.

That this procedure results in the new updated shortest path matrix can be seen from the following argument:

1) There is nothing inherently special in the numbering of the nodes in the algorithm, hence we can renumber the nodes arbitrarily, e.g., one can rename node 1 as node 6 and node 6 as node 1.

2) Applying Step 2 of Floyd's algorithm to J' for any i not equal to x or y will not change J' , because Step 2 will change J' only if elements in row i or column i have been changed (which is only true for $i=x$ or $i=y$). This is because J was already a shortest path matrix, and hence applying Step 2 to J for *any* i will not change J .

3) As there is nothing inherently special about the numbering of the nodes, we can renumber the nodes such that node x becomes node $n-1$ and node y becomes node n .

4) Applying Floyd's algorithm to this renumbered matrix will produce no changes for the first $n-2$ iterations; the only ones that will produce any changes are the last two iterations.

5) Reordering the rows and columns of a matrix will have no effect on Step 2 (the generalized outer product with respect to addition), except that the resulting matrix will also be similarly reordered. Hence it is not really necessary to run Floyd's algorithm incrementing i by 1 at each iteration; all that is really necessary is that on each iteration i be chosen as some integer between 1 and n without duplication. Therefore it is also not necessary to renumber the nodes; the shortest path matrix can be updated directly. The two iterations of Floyd's algorithm require n squared computations.

A.5 Implementation Considerations

In implementation of the crew activity planning problem, constraints are added interactively by the user, and not in a batch. Hence, the shortest path matrix can be updated after each constraint is added, with n squared computations. For reasonable problems, this amount of computation can be done "in real time" so that a user can immediately see the effects of each new constraint on the shortest path matrix (and hence on earliest and latest start times). It is true that in the worst case, there can be n squared constraints, thus resulting in the order of n to the fourth computations, but in practice there are far fewer constraints (and the delay for each constraint addition in an interactive environment is negligible anyway).

In addition to maintaining the Δ MAX matrix, it is also necessary to maintain a record of the latest end times and the precedence constraints. This is because these constraints contain more restrictions upon the schedule than is included in their transformation to Δ MAX constraints, as described previously.

In order to maintain the latest end time constraints, a vector LET can be constructed, where $LET(x)$ is the latest end time of task x . This vector is initialized to $LET(x)=ME$ and then each entry is replaced by any explicitly entered value for $LET(x)$ whenever this value is less than the current value. Inference can also be used to infer a reduced value for $LET(x)$ according to the formula: $LET(x)=\text{MIN} \{LET(x), LST(x)+MAXT(x), LST(y) \text{ for all } y \text{ such that } P(x,y)=1\}$. With interactive entering of constraints, this calculation may have to be performed after each new constraint is entered, because any new constraint can potentially impact the latest start times of some relevant task. By requiring that each task start after $EST(x)$, start before $LST(x)$, and end before $LET(x)$ (as found from the $\Delta MAXS$ matrix and the LET vector) the full inference from the constraints is fully realized.

As tasks become scheduled, one can then make further inference based on the now known start and end times of the task. For example, suppose task x becomes scheduled with start time ST and end time ET . Note that $ST \geq EST(x)$, $ST \leq LST(x)$, $ET \leq LET(x)$, and $MINT(x) \leq (ET-ST) \leq MAXT(x)$. If $(ET-ST) > MINT(x)$, we can then make a greater inference from $P(x,y) = 1$ than $\Delta MIN(x,y) = MINT(x)$. This is effectively done by changing $MINT(x)$ (and $MAXT(x)$) to $ET-ST$ and then adding and propagating new constraints $\Delta MIN(x,y) = ET - ST$ (i.e. $\Delta MAX(y,x) = -\mu$) for all y such that $P(x,y) = 1$. We also add and propagate new (tighter) constraints $EST(x) = ST$, $LST(x) = ST$, and $LET(x) = ET$. While the propagation of the constraints added as a result of scheduling task x may cause the windows for other tasks to tighten, it cannot cause them to become infeasible: if this were not true, we could have made further inferences from the performance times and constraints on the other tasks, thus making the window for task x smaller - but the window for task x is already as small as can possibly be inferred from the other jobs - a contradiction.

Through active constraint propagation, it is thus possible to maintain values for the earliest and latest possible start times and the latest possible end time for each task. Further, one can check, as each constraint is entered, that the constraint does not cause some diagonal element of $\Delta MAXS$ to become negative, thus implying a logical contradiction. The time windows thus obtained by this algorithm guarantee that all tasks can be sequenced within their planning windows so that they observe all time constraints. One can further guarantee that if a task is scheduled to start after its earliest start time and before its latest start time, and to end before its latest end time, as determined from constraint propagation, then it cannot possibly force some other task to become infeasible (i.e. not be schedulable within its time constraints). Of course, it is still possible to have an infeasible

schedule because of conflicts caused by resources limits or a finite number of processors (crewmembers), but this algorithm goes a long way towards reducing the complexities of avoiding infeasibilities.

A.6 Storage Considerations

The disadvantage of Floyd's algorithm is that it requires storage of n squared numbers. For n on the order of 1000, as in the crew activity scheduling problem, this requires on the order of 1,000,000 entries. It should be noted that the only results that are really needed are the $2n$ entries in the first row and first column of ΔMAXS . Dijkstra's algorithm, on the other hand, has little stowage requirement beyond the explicit list of constraints and the initialized values of $\Delta\text{MAX}(*,x)$ and $\Delta\text{MAX}(x,*)$. Dijkstra's algorithm, as modified to allow for arcs of negative length, requires order of n cubed computations for finding the shortest path from any given node to all other nodes, and it must be used twice: once to find the shortest path from the source node (*) to all other nodes, and once to find the shortest path from all nodes to the source node (this is done by reversing the directions of all the arcs and then finding the shortest path from the source to all other nodes). There are several other disadvantages to Dijkstra's algorithm: it is difficult to program, and there is no analagous way to update the shortest paths as with Floyd's algorithm; adding a new arc can necessitate repeating the entire algorithm.

Floyd's algorithm can be modified to require less storage provided that the problem is decomposable. The problem can be decomposed if groupings of tasks can be found for which the constraints on each of the tasks in the group pertain only to * and not to other tasks in other groups (this will often be the case, as the ΔMAX matrix will usually be sparce of significant information other than in the * row and column). Consider the case where there are 5 tasks in addition to *, and where there are constraints relating task 1, 2, and 3, and other constraints relating only tasks 4 and 5. Then all that is required is two matrices, one 4 x 4 (containing *, 1, 2, and 3) and the other 3 x 3 (containing *, 4, and 5). This requires a storage of 25 numbers as opposed to the 36 numbers required for the full 6 x 6 matrix. Savings can be much greater by decomposing larger matrices, although in the worst case all tasks will be interrelated, and the full $n \times n$ matrix is necessary.

To implement this modification, one would start with a 2 x 2 matrix for each task (containing * and the task). Constraints relating the tasks to * can then be added directly to the task's matrix.

When constraints relating tasks to other tasks are processed or added, new matrices are formed by merging the two 2×2 matrices for the two tasks, thereby creating a 3×3 matrix. Henceforth, when any constraints are found relating these same tasks, the matrix can be updated directly (note that this updated computation is now on the order of the square of the size of this smaller matrix, not the order of n squared). Whenever constraints relate tasks in different matrices, the matrices are merged, as above. It should be noted that the storage requirements can be further reduced by not storing the diagonal elements in the original 2×2 matrices, as they are zero.

Merging the matrices is accomplished in a straightforward manner. Consider the previous example with one matrix containing tasks *, 1, 2, and 3 and the other matrix containing tasks *, 4, and 5. Adding a constraint relating tasks 2 and 5 will necessitate merging the two matrices into a 6×6 matrix containing tasks *, 1, 2, 3, 4, 5, and 6. The positions for which there are no data, e.g. (2,4), are set to infinity (or some suitably large number). Then to update the matrix, all one has to do is run Step 2 of Floyd's algorithm three times: once with $i=*$ and once each for i equal to the two tasks of the new constraint (in this case $i=2$ and $i=5$). Merging is thus also an order of n squared computation.

Bibliography

- Balbaky, E.M.L.: Strike in Space. Case #1-431-008, Harvard Business School, Boston Massachusetts, 1980.
- Bensana, E.; Corregge, M.; Bel, G.; Dubois, D.: An Expert-System Approach to Industrial Job-Shop Scheduling. Proceedings of the 1986 IEEE International Conference on Robotics & Automation, April 7-10, 1986, pp. 1645-1650.
- Blagov, V.D.: Deputy Flight Director Blagov on 211-Day Flight of Berezovoy and Lebedev. USSR Report: Space, June 14, 1984. JPRS-USP-84-003.
- Blazewicz, J.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Scheduling Subject to Resource Constraints: Classification and Complexity. Stichting Mathematisch Centrum, Amsterdam, Department of Operations Research, August 1980.
- Chang, S.M.; Sielken, R.L., Jr.: Evaluation of Precedence Criteria for Project Scheduling Under Resource Constraints. Texas A & M University, Project Themis, Technical Report #63, May 1980.
- Conway, R.W.; Maxwell, W.L.; Miller, L.W.: Theory of Scheduling. Addison-Wesley, Reading, Massachusetts, 1967.
- Cooper, H.S.F., Jr.: A House in Space. Holt, Rinehart and Winston, New York, 1976.
- Davis, E.W.; Patterson, J.H.: A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. Management Science, Vol. 21, No. 8, April 1975, pp. 944-955.
- Dempster, M.A.; Lenstra, J.K.; Rinnooy Kan, A.H.G.; eds.: Deterministic and Stochastic Scheduling. NATO Advanced Study Institutes Series. Series C: Mathematical and Physical Sciences, Vol. 84. D. Reidel, 1982.
- Engelman, C.; Millen, J.K.; Scarl, E.A.: KNOBS: An Integrated Planning Architecture. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983, pp. 450-462.
- Erschler, J.; Esquirol, P.: Decision-Aid in Job Shop Scheduling: A Knowledge Based Approach. Proceedings of the 1986 IEEE International Conference on Robotics & Automation, April 7-10, 1986, pp. 1651-1656.
- Fisher, M.L.; Jaikumar, R.: An Algorithm for the Space-Shuttle Scheduling Problem. Operations Research, Vol. 26, No. 1, January-February 1978, pp. 166-182.
- Fox, M.S.; Allen, B.P.; Smith, S.F.; Strohm, G.A.: ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling. Intelligent Systems Laboratory, Carnegie-Mellon University, Pittsburgh, PA, June 21, 1983.
- Freedman, D.H.: PC Software Helps Projects Run Smoothly. High Technology, May 1985.

- Gevarter, W.B.: An Overview of Expert Systems, NASA Headquarters, May 1982. NBSIR 82-2505.
- Glazer, S.: Infocel: Two Start-ups are Better than One. High Technology, May 1985, p. 20.
- Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Ann. Discrete Math., Vol. 5, pp. 287-326.
- Grenander, S.: Toward the Fully Capable AI Space Mission Planner. Aerospace America, Vol. 23, No. 8, August 1985, pp. 44-46.
- Grone, R.D.; Mathis, F.H.: A Ranking Algorithm for Spacelab Crew and Experiment Scheduling. 1980 NASA/ASEE Summer Faculty Research Fellowship Program. N81-11986, October 1980.
- Hankins, G.B.; Jordan, J.W.; Katz, J.L.; Mulvehill, A.M.: Expert Mission Planning and Replanning Scheduling System. The MITRE Corporation, Bedford, Massachusetts, September 1985.
- Hayes-Roth, F.; Waterman, D.A.; Lenat, D.B.; eds.: Building Expert Systems. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1983.
- Hitz, F.R.: Payload Crew Activity Planning Integration. Task 2: Inflight Operations and Training for Payloads. Contract NAS9-14676. Martin Marietta Corp., Denver, Colorado, N77-18739, December 23, 1976.
- Ivakhnov, A.: Changes in 'Soyuz T-10' Crew Work Schedule. USSR Report: Space, June 14, 1984, p. 18. JPRS-USP-84-003. Translated from Izvestiya, Feb. 17, 1984.
- Jacobowicz, O.: Automatic Mission Planning & Scheduling Expert System (AMPASES). Status Report, GE/TRW, October 30, 1985
- Jardine, T.J.; Shebs, S.T.: Knowledge Representation in Automated Mission Planning. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983, pp. 9-14.
- Johnson, R.D.; et al: Autonomy and the Human Element in Space: Final Report of the 1983 NASA/ASEE Summer Faculty Workshop, NASA, 1985.
- Kranzler, D. A.: An Integrated Stowage Logistics Clerk For Space Station Autonomy. Bachelor of Science Thesis for the Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, Massachusetts, June 1986.
- Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G.: Recent Developments in Deterministic Sequencing and Scheduling: A Survey. Stichting Mathematisch Centrum, Amsterdam, Department of Operations Research, August 1981.

- Litsov, A.N.; Bulyko, V.I.: Principles of Organization of Rational Schedules for Crew Work and Rest During a Long-Term Spaceflight. USSR Report: Space Biology and Aerospace Medicine, Vol. 17, No. 4, July-August 1983, pp. 9-13.
- Marsh, A.K. (1984a): NASA to Demonstrate Artificial Intelligence in Flight Operations. Aviation Week & Space Technology, September 17, 1984, Vol. 121, No. 12, p. 79.
- Marsh, A.K. (1984b): Pace of Artificial Intelligence Research Shows Acceleration. Aviation Week & Space Technology, December 10, 1984, Vol. 121, No. 24, pp. 24-25.
- Mathis, F.H.: Mathematical Programming Techniques for Scheduling Spacelab Crew Activities and Experiment Operations. NASA/ASEE 1981 Summer Faculty Research Fellowship Program. N82-17075, August 14, 1981
- Mogilensky, J.; Scarl, E.A.; Dalton, R.E.: Manned Spaceflight Activity Planning With Knowledge-Based Systems. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983.
- Murphy, W.W.; Krusemark, K.A.; Moyer, R.W.: Increased Crew Activities Scheduling Effectiveness Through the Use of Computer Techniques. Human Factors, Vol. 10, No. 1, February 1968, pp. 57-62.
- Pritsker, A.A.B.; Watters, L.J.; Wolfe, P.M.: Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. Management Science, Vol. 16, No. 1, September 1969, pp. 93-108.
- Riley, V.; Vestewig, R.: A Voice Interactive System for Aiding and Documentation of Space-Based Tasks. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983, pp. 171-177.
- Scarl, E.A.; Engelman, C.; Pazzani, M.J.; Millen, J.: The KNOBS System. The MITRE Corporation, Bedford, Massachusetts, undated.
- Smith, S.F.: Exploiting Temporal Knowledge to Organize Constraints. Intelligent Systems Laboratory, Carnegie-Mellon University, Pittsburgh, PA, July 19, 1983.
- Space Station Reference Configuration Description. Systems Engineering and Integration, Space Station Program Office, NASA, Lyndon B. Johnson Space Center, Houston, TX, 1984.
- Stein, K.J.; et al: Boeing Accelerates Research, Dissemination of Technology. Aviation Week & Space Technology, February 17, 1986, Vol. 124, No. 7, pp. 71-79.
- Talbot, F.B.; Patterson, J.H.: An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained-Scheduling Problems. Management Science, Vol. 24, No. 11, July 1978, pp. 1163-1174.

- Townsend, W.B.: Artificial Intelligence Techniques for Industrial Applications in Job Shop Scheduling. Naval Postgraduate School, Monterey California. M.S. Thesis, June 1983. AD-A132-164.
- Vere, S.A. (1983a): Planning in Time: Windows and Durations for Activities and Goals. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAM1-5, No. 3, May 1983.
- Vere, S.A. (1983b): Planning Spacecraft Activities with a Domain Independent Planner. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983.
- Vere, S.A.: Deviser: an AI Planner for Spacecraft Operations. Aerospace America, Vol. 23, No. 4, April 1985, pp. 50-53.
- Wagner, R.E.: Expert System for Spacecraft Command and Control. AIAA Computers in Aerospace Conference, Hartford, CT, October 1983, pp. 216-223.
- Weiss, S.M.; Kulikowski, C.A.: A Practical Guide to Designing Expert Systems. Rowman & Allanheld, NJ, 1984.