AN EXPERT SYSTEM FOR DYNAMIC SCHEDULING

DONNIE FORD and STEVE FLOYD
THE ARTIFICIAL INTELLIGENCE LABORATORY
JOHNSON RESEARCH CENTER
UNIVERSITY OF ALABAMA - HUNTSVILLE
HUNTSVILLE, AL 35899

ABSTRACT Traditionally, scheduling problems have been viewed as static in nature (i.e., a schedule is developed for a particular planning horizon and adhered to) and were cast as having one or more clearly defined objectives (e.g., minimize overall completion time, maximize resource utilization, etc.). These problems were most commonly solved via application of optimal seeking algorithms, heuristics or simulation analysis [1] [5] [9] [10] [17]. The payload scheduling problem is representative of a class of scheduling problems which are highly dynamic in nature. That is, the various parameters can change at any time, and the objectives themselves may change also. The nature of this class of problems is such that they can be most effectively solved by knowledge based expert systems [2] [3] [8] [13] [19] [20].

KEYWORDS: Expert systems, knowledge-based, heuristic, scheduling, dynamic, prototype

PROBLEM DESCRIPTION

The problem under investigation in this research concerns the scheduling of subsystems and payloads aboard the space station. Subsystems are systems which function to support space station on an ongoing basis. These include such subsystems as life support systems, communications systems, and various "housekeeping" systems. Aboard space station will also be various payloads and experiments. These will be sponsored not only by NASA but also by other U.S. and foreign government agencies, universities and private industries.

Each of the subsystems or payloads has a certain set of characteristics and requirements which must be considered in determining when during the mission it should be scheduled. For example, each subsystem and most of the payload/experiments will draw operating power from Space Station's limited power supply. Additionally, certain of them will require astronaut intervention either on a continuous basis for the duration of the experiment or for specified subintervals of time. Some subsystems and experiments are continuous in nature and run uninterupted throughout the entire mission. Still others operate either continuously or intermittently for only a specified subinterval of the mission time window. The nature of some experiments will require that they be conducted only during certain phases of the mission (e.g., during day orbit, during night orbit, during certain orientations of space station, etc.). These example characteristics, as well as others which will not be detailed here, coupled with the fact that payload/experiments are

placed in priority classes which must be reflected in the schedule, form the basic criteria for establishing feasible schedules.

The complexity of the scheduling problem is compounded further by the fact that events which will be occurring during the mission will serve either directly or indirectly to upset current schedules and/or influence future ones. For example, at any time during the mission an ongoing experiment may fail or be aborted for some reason, a scheduled experiment may be withdrawn from the schedule, an experiment or entire class of experiments may be added and/or experiment priorities changed. The scheduler must be designed to handle such dynamic changes.

As mentioned previously, each subsystem and payload/experiment will consume various resources. Major among these will be energy from the Space Station's power supply and manpower provided by the astronauts on board. Such limited resources place constraints on what systems and experiments can be concurrently ongoing. Additionally, and this is another of the dynamic aspects of the problem, the power and manpower allotments themselves may change at various times throughout the mission. In some instances the change notification will provide lead time for scheduling adjustments, whereas in others no lead time will be provided. Changes will occur, for example, when vehicles dock with Space Station. Such changes result from the fact that the docking will usually draw on such resources as the power and manpower supply. In light of the above mentioned characteristics, the scheduler must have capabilities beyond the generation of traditional static feasible schedules. The dynamic scheduler must have the capacity to respond to such changes and, when required, maintain feasibility via a rescheduling procedure.

The final characteristic of the class of problems to which this research relates is the dynamic nature of the objectives. During the course of a Space Station mission, the structure of the scheduling objectives may change. For example, it might be that early in a mission the most important scheduling objective is maintenance of a fairly constant and conservative power consumption rate. Such an objective would naturally "stretch out" the scheduling of experiments over some designated planning horizon. Later in the mission cycle, however, factors may change this objective to one of scheduling as many payloads/experiments as possible (subject to the maximum power availability and other constraints) in a given time frame. These characteristics then establish the need to develop a system which is capable of not only establishing but also of maintaining feasible payload/experiment schedules which reflect the varying parameters of the problem.

Sample data around which a prototype system could be constructed was provided by NASA. The data was considered by NASA to be representative of actual scheduling data, four subsystems and forty-five (45) payloads/experiments were included. Also, provided are the experiment name, the associated power consumption requirements in kilowatts, the sponsoring agency, the time duration (including other specifications such as continuous/intermittent, day orbit/night orbit, etc.) and crew involvlement required. In addition to the data, other problem specifications were provided. Most pertinent among these were (1) the specification of a normal lab module power level of 25 kilowatts, (2) a priority structure based on the sponsoring agency and the nature of the payloads/experiments, and (3) a two-week scheduling horizon. Additionally, several system requirements pertaining to the actual operation of the scheduler were specified. These provided a framework for the user

interface and system output as detailed later in the system description section of the paper.

The traditional O.R. techniques for scheduling applications are simulation, network methods, combinatorial procedures, and heuristic approaches. The choice of technique usually depends on the problem complexity, the type of the model, the choice of objective, and other factors.

Network methods are inapplicable to dynamic scheduling because the precedence network is constantly changing. Combinatorial procedures can be ruled out, because of the complexity of the problem.

The only remaining computer-based techniques that are applicable are simulation techniques and heuristic approaches. However, simulations have to be interpreted by skilled O.R. scientists before a naive user can readily understand them. Also, the "cycle time" for simulation is too long. Existing heuristic-scheduling programs have limited intelligence, because of the very simple knowledge representations used. Therefore, an A.I.-assisted heuristic-scheduling program seemed to be the only workable approach.

There are several A.I. scheduling programs currently in existence. Some of the programs are as follows:

NUDGE/BARGAIN was developed at MIT by Ira Goldstein, et.al.[6]

ISIS-II was developed at Carnegie-Mellon University by Mark Fox, et.al.[4]

NONLIN was developed at Edinburgh University, England by Austin Tate[16]

DEVISER was developed at the Jet Propulsion Laboratory by Stephen Vere[17]

FIXER is being developed at Brunel University, Uxbridge, England by T. J. Grant. Also, another knowledge-based program for scheduling a V.L.S.I. Wafer Fabrication Laboratory is being developed using an approach similar to that of FIXER[7].

SOLUTION STRATEGY

In developing a prototype system, a means of generating feasible schedules in the absence of a fully developed knowledge base had to be developed. This was accomplished via the use of a scheduling heuristic. The heuristic was developed by conceptualizing schedules using a Gantt chart format. An example schedule for a simple four experiment problem is given in figure 1. As can be seen, experiments one, two and three are continuous, and experiment four is intermittent. Given inside the bars, which represent the experiment durations, is the power requirement of the particular experiment. For simplicity these power requirements are assumed constant as long as the experiment is "on." Through the use of this four experiment example, the heuristic procedure will now be described.

As an experiment is placed on the chart, its beginning and ending point(s) serve to divide the overall time window into intervals as illustrated by the dotted lines in figure 1. By updating as each experiment is scheduled, one can maintain for each subinterval of time the information necessary in determining the time slot for the next experi-

ment to be scheduled. The determination of which experiment is to be scheduled next is based on the priority structure in effect at the time of scheduling. To simplify the four experiment example further, we will assume a single scheduling objective of maximizing power utilization. Each experiment is scheduled by searching the time axis in figure 1 from left to right until a subinterval or group of successive subintervals is found which has sufficient time duration and power availability to support the given experiment. The experiment is then scheduled and added to the chart in correspondence with this subinterval. Subinterval information is updated accordingly, and the scheduling procedure continues.

Applying the heuristic to the representative problem provided by NASA is obviously much more involved than the example provided above as the various experiment characteristics and requirements must be matched to appropriate intervals. As the number of requirements increases for experiments, so too does the amount of information being kept on each subinterval. Additionally, as the number of experiments already scheduled increases, the number of subintervals to be examined during each individual scheduling process also increases. This increase in the number of subintervals is compounded even further when the experiment scheduled is of an intermittent nature. These facts, coupled with the previously mentioned dynamic aspects of the problem, necessitate an automated procedure for generating schedules. The next section of the paper will describe the system developed to accomplish this.

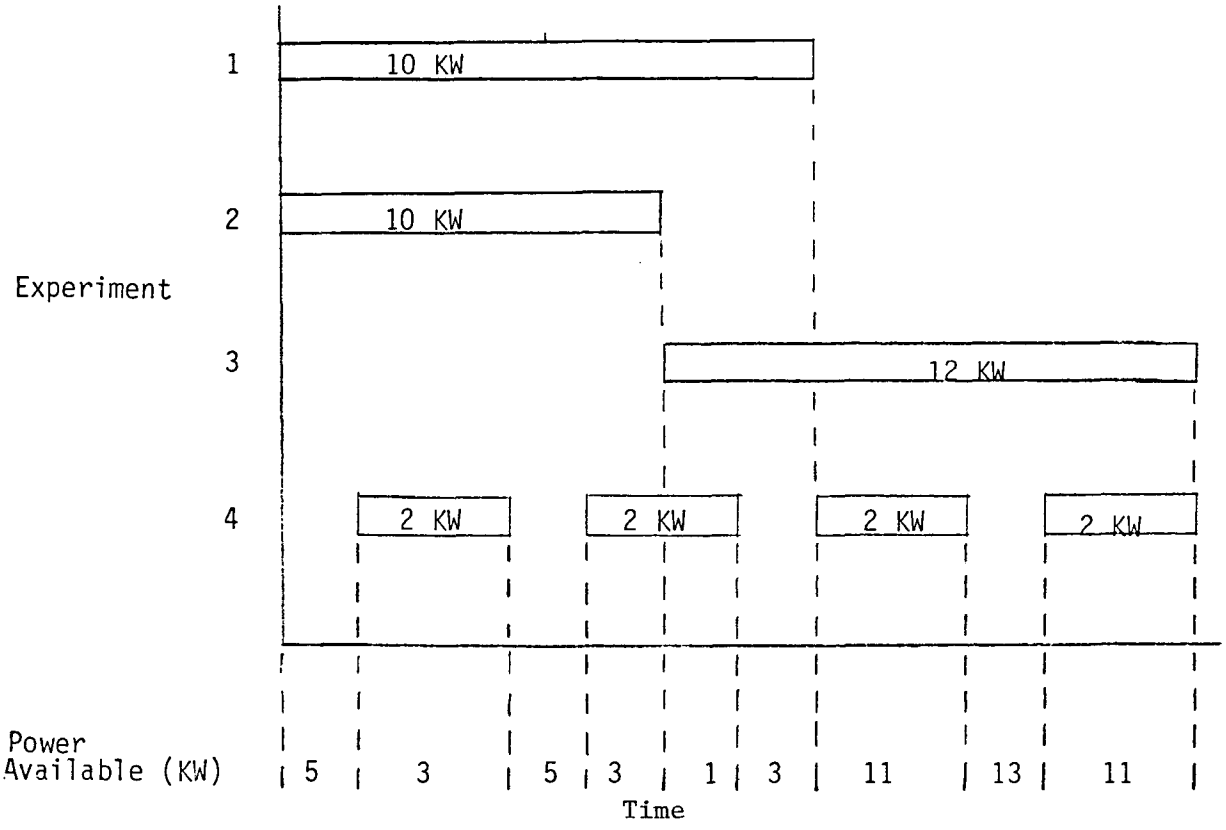

Figure 1. Example Schedule

SYSTEM DESCRIPTION

The above strategy was internalized using Zetalisp on a Symbolics 3670 Lisp machine [15]. AI development tools were not utilized on this project. The system follows the basic production system structure of a knowledge base, inference engine and working memory or global data base. The knowledge base consists of the scheduling rules and other knowledge pertinent to the problem. The system utilizes frame representation which allows for utilization and exploitation of knowledge other than rules. This feature increases the speed and efficiency of the system, in particular, the inferencing process.

The inference engine performs only forward chaining. This was determined from the structure of the problem. There is an abundance of related facts and information at the beginning of the problem solving process which in turn accomodates the forward chaining process. Also, the number of acceptable goal states is indeterminate. The conflict resolution problem is solved by allowing the first rule that is satisfied to be implemented. This necessitates an ordering of the rules. This resolution method was chosen because of the short time frame for delivering a "demo" system. This also facilitates the search through the working memory.

The working memory consists of a list of experiment names. Associated with these names are certain facts that are placed into the knowledge base. These include the power requirements, identification number, priority class, sponsoring agency, duration of the experiment, and the required crew involvement. Using this information, a prioritized list of experiments is generated for utilization during the scheduling phase.

The system has been designed and developed in an open-ended fashion to allow the system to be extended with only minor adjustments to the system. It contains, in addition to the expert system structure, an output interface which is for demonstration purposes at present. This interface will be detailed later in the paper. The system itself is dynamic in that it moves through or between different phases of the problem. The phases include preparation, scheduling, operation, and rescheduling.

During the preparation phase the individual experiment information is appropriately stored, and the working memory is organized and then prioritized for the scheduling phase. In the scheduling phase, the experiments are scheduled under the previously explained heuristic procedure and the schedule is created. The schedule itself is part of the knowledge base and is represented as frames. The schedule is incremented in minutes and extends over a time horizon of two weeks as was specified in the problem definition. As experiments are scheduled, the subintervals required by the heuristic procedure are defined by start and stop times of the experiments. For each interval the power available, crew available, and the experiments that are on are determined and stored. This information is required for the remaining two phases, operation and rescheduling.

The output interface during the operation and rescheduling phases is graphical in nature and menu driven. This facilitates the use of the system as the user is not required to know the syntax of the LISP language [21]. During the operation and rescheduling phases, the system actually controls the experiments, i.e., it turns them off or on at the

appropriate times and updates all the necessary interface information accordingly. The operation phase has two modes: (1) static and (2) dynamic. In the static mode, the system is capable of displaying a power utilization graph for a two week, one week, one day or six hour period of time. Also, the vital information for each experiment (start time, end time, etc.) can be displayed simply by using the mouse and a selection menu. In the dynamic mode, the system uses the output interface to display four basic windows: a current status window, a schedule window a power curve window and a message window. The current status window shows the current status of all the experiments of a payload at a particular point in time. This is accomplished, by representing each experiment as a numbered window. Reverse video is then used to differentiate the on state and labels placed within the boxes indicate such statuses as aborted , removed, completed, etc. The schedule window displays the names of the experiments on separate lines and uses a Gantt chart format similar to that shown in figure 1 to display the scheduling of each experiment. This window moves through time, i.e. as time passes the bars that represent the experiment move to the left and disappear as the experiment is completed. When the experiment is completed the word completed appear next to the experiment name. In the schedule window the experiments are also numbered to provide a cross-reference to the numbered experiment boxes in the current status window. The power curve window (middle right of display screen) plots percent power utilization as it scrolls through time. The remaining window is the message window. This is used for control purposes only.

One of the important capabilities built into this system is its ability to reschedule the experiments when deemed necessary. This is one of the main differentiators of this system when compared to others developed for such scheduling applications. The system is capable of determining when it is necessary to reschedule. When such a determination is made, the experiments affected are identified and removed from the active schedule. A rescheduling is conducted and the new schedule is implemented (i.e., made active). There are, based on the initial problem description, a limited number of occurences that would warrant a reschedule. These include an experiment failure, an experiment abort, a power allotment increase or decrease, or the announced arrival of orbital docking and/or serving vehicles. The first two occurrences require an automatic rescheduling while the others require the system to check working memory and the knowledge base to determine if rescheduling is in fact necessary. Thus we see the system is capable of moving between the different phases, capable of recognizing where it is and what knowledge is applicable, and dynamic in its ability to genenrate and maintain a schedule that will accomplish the objective or objectives of the mission.

FUTURE ENHANCEMENTS

While the system demonstrates the potential of using the expert system approach in the area of scheduling, there are several enhancements that have been identified to improve the performance and capabilities of the system. First and foremost is that more experiential knowledge needs to be added to the knowledge base. Sessions have been scheduled with the appropriate NASA personnel to begin the task of knowledge engineering [2] [3] [9] [13]. Also, knowledge concerning the determination of alternatives to the schedule instead of just developing

a single initial schedule will be added. This will provide the system with the capability of helping NASA personnel in satisfying the dynamic objectives experienced during a mission and will also facilitate the rescheduling process. An example of such an objective is when power allotment reduction forces the schedule to run past the end-of-mission time. Having "knowledge" of alternatives, the system will have a better understanding of which experiments to schedule. Should it continue with the normal rescheduling rules or does some special set of priorities apply? Not only will more objectives be handled, but the system will be able to handle more constraints, e.g., fluctuating power requirements of experiments, and orientation of the experiments. These are just two of the various constraints which apply during an actual mission. Another area of knowledge enhancement is in the area of rescheduling. It has been determined that the system should be capable of performing a quick-fix reschedule when necessary. This will provide the system the necessary time to perform a more detailed and thorough reschedule in the event of an emergency situation where a quick decision is necessary. This particular area is being studied in more detail at present.

The second area of improvement and enhancement to the system is efficiency. Not only is the efficiency of the code being considered, but a more efficient and effective method for searching the schedule and determining experiment slots is under development. This search process, as was mentioned previously, is complicated by the scheduling of intermittant experiments early in the scheduling process. In particular, one experiment in the sample data is required to be scheduled ten minutes out of every hour that the mission is operating. Under the present system this creates 336 additional time intervals that might have to be checked for power and crew availability in determining a feasible interval for a later experiment. The present system takes 15 minutes to schedule the 47 test experiments. The majority of this time is due to the intermittent experiments. Another efficiency enhancement is for the system itself to determine the mission time horizon. This will reduce any excess time that is added in order to accomodate all the experiments. At present, the time horizon is driven by the number of experiments requireing crew involvement and the number of crew available to work with the experiments. A critical path algorithm is being investigated for application in this area.

The final area of enhancements is in the user interface, both input and output interfaces. On the input side, a query/answer system will be added to the system to allow for easy input of experiment data and knowledge base maintenance. This interface will have a limited, natural language parser [12] [14] and will exploit the use of graphics. On the output side several enhancements will be made. First, the system will have an explanation capability to explain how and why it chose the schedule it is recommending. Second, a graphical display of the entire schedule in Gantt chart format will be added to the static mode. Third, a hardcopy capability for printing out the schedule in "readable" form will be added. Currently the schedule is only stored in symbiolic form. These output enhancements will facilitate the evaluation of system performance. This should allow the system in turn to gain user acceptance more quickly and will also help facilitate the implementation phase. The final item to be accomplished is the actual linking of the system and its host machine to the control circuits of experiments.

CONCLUSIONS

This paper has detailed a knowledge-based system for solving the NASA space station payload/experiment scheduling problem. The problem is representative of a larger class of dynamic scheduling problems which, for the most part, have been ineffectively handled using more traditional numeric techniques. An expert systems approach allows one to effectively deal with the dynamics and incomplete information which characterize this class of problem. Still in prototype form the system is meeting with wide acceptance and interest not only from the sponsoring agency, but also from other independent sources.

The interest this project has received indicates that there is potential for further research in this area. The wide problem domain encompassed by dynamic scheduling provides many areas for future applications (e.g., project scheduling, production scheduling, manpower scheduling, etc.). Additionally, as systems are implemented and knowledge engineering continues, there is a good likelihood that commonalities will be established across various scheduling applications. This would allow devlopment of an expert system shell for such problems. Such a shell would allow scheduling systems to be readily developed and implemented.

REFERENCES

1. Baker, K. R., Introduction to Sequencing and Scheduling, New York: John Wiley and Sons, Inc. 1974.

2. Barr, A. and Fiegenbaum, E., The Handbook of Artificial Intelligence, Volumes I and II, William Kaufmannm Inc., 1982.

3. Davis, Randall, and Douglas B. Lenat, Knowledge - Based Systems in Artificial Intelligence, New York: McGraw - Hill, Inc., 1982.

4. Fox, Mark S., B.P. Allen, and G.A. Strohm, "Job-shop Scheduling: An Investigation in Constraint-Directed Reasoning." Proceedings of AAAI-82, Carnegie-Mellon University, Pittsburgh, PA., 1982.

5. French, S., Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop, New York: John Wiley and Sons, Inc.,1982.

6. Goldstein, I.P. and R.B. Roberst, "NUDGE a Knowledge-based Scheduling Program." IJCAI 5, (1977), pp. 257-263.

7. Grant, T.J., "Lessons for O.R. From A.I.: A Scheduling Case Study." Journal for the Operational Research Society, Vol. 37 No. 1, 1986, pp. 41-57.

8. Harmon, Paul, and David King, Expert Systems, New York: John Wiley and Sons, Inc., 1985.

9. Johnson, L. A. and D. C. Montgomery, Operations Research in Production Planning, Scheduling and Inventory Control, New York: John Wiley and Sons, Inc., 1974.

10. Lawler, E. L., J. K. Lenstra and A. H. G. Rinnooy Kan, "Recent Developments in Deterministic Sequencing and Scheduling: A Survey," in M. A. H. Dempster et. al. (Eds.), Deterministic and Stochastic Scheduling, Reidel, Dordrecht, 1982, 35-73.

11. Phelps, R.I., "Artificial Intelligence-An Overview of Similarities With O.R." Journal of the Operational Research Society, Vol. 37 No. 1, 1986, pp. 13-20.

12. Rauch-Hindin, Wendy, "Natural Language: An Easy Way to Talk to Computers," Systems & Software, January, 1984, pp. 187-230.

13. Riesbeck, C. K. and Roger Schank, "Comprehension by Computer: Expectation-based Analysis of Sentences in Context," in W. J. M. Levelt and G. B. Hores d'Arcais (Eds.),Studies in the Perception of Language. Chichester, England: John Wiley and Sons, 1976, pp. 247-294.

14. Rich, E., Artificial Intelligence, New York: McGraw-Hill, Inc., 1983.

15. Symbolics software. Report, Symbolics, Inc., 21150 Califa Street, Woodland Hills, California, 1981.

16. Tate, Austin, "Project Planning Using a Hierarchical Nonlinear Planner." Report No. 25, A.I. Research Department, university of Edinburgh, 1976.

17. Tersine, Richard J., Production/Operations Management: Concepts, Structure, and Analysis, (2nd ed.), New York: North-Holland Press, 1985.

18. Vere, Stephen, "Planning in Time: Windows and Durations for Activities and Goals." I.E.E.E. Pattern Annual Mechanical Intelligence., PAMI-5, 1983, pp. 246-266.

19. Waterman, Donald A., A Guide to Expert Systems, Reading, Massachusetts: Addison-Wesley Publishing Company.

20. Winston, Patrick H., Articial Intelligence, (2nd ed.), Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

21. Winston, P. H., and Horn, B. K. P. LISP, Reading, Massachusetts: Addison-Wesley Publishing Company.