SLS-PLAN-IT: A Knowledge-Based Blackboard Scheduling System for Spacelab Life Sciences Missions

Cheng-Yan Kao (+)
Dept. of Computer Science
& Information Engineering
National Taiwan University
Taipei, Taiwan 107

Tel: 886-2-363-0231 ext. 3231
Fax: 886-2-362-8167

&

Seok-Hua Lee (*)
GE Government Services
General Electric Corporation
1050 Bay Area Blvd
Houston, TX 77058, USA

Tel: (713) 488-9005
Fax: (713) 488-1092

ABSTRACT

The Mission Integration Office (MIO) of GE Government Services was responsible for generating and updating the crew activity plan and resource assignments for the Spacelab Life Science SLS-1 mission for NASA. The nine-day SLS-1 shuttle mission was launched on June 5, 1991.

The Spacelab mission planning was an overconstrained domain. There were over fifty resources and several hundred activities with several thousand steps to be scheduled in the SLS-1 mission. This is an NP-hard problem. The primary scheduling tool in use during the SLS-1 planning phase was the operations research (OR) based, tabular form Experiment Scheduling System (ESS) developed by Marshall Space Flight Center (MSFC).

PLAN-IT is an artificial intelligence (AI) based interactive graphic timeline editor for ESS developed by Jet Propulsion Laboratory (JPL). We have enhanced the PLAN-IT software for use in the scheduling of Spacelab experiments to support the Spacelab Life Science missions. The enhanced software SLS-PLAN-IT System was used to support the real time reactive scheduling task during the SLS-1 mission. This software will be further enhanced before the SLS-2 mission and is expected to completely replace the ESS currently in use in MIO in the SLS-3 time frame.

SLS-PLAN-IT is a frame-based blackboard scheduling shell which, from scheduling input, creates resource-requiring event-duration-objects and resource-usage-duration-objects. The blackboard structure is to keep track of the effects of event-duration-objects on the resource-usage-duration-objects. The constraints are propagated automatically for conflict resolution. Various scheduling heuristics are coded in procedural form and can be invoked any time at the user's request. The timeline entries can be manipulated by the mouse to support the scheduling task. This paper describes the system architecture and what we have learned with the SLS-PLAN-IT project.

(+) The first author was involved in this project when he was an employee of GE Government Services, Houston, Texas.
(*) All correspondence should be sent to the second author.

Introduction:

The Mission Integration Office (MIO) of GE Government Services was responsible for generating and updating the crew activity plan and resource assignments for the Spacelab Life Science SLS-1 mission for NASA. The nine-day SLS-1 shuttle mission was launched on June 5, 1991. The primary scheduling tool in use during the SLS-1 planning phase was the Experiment Scheduling System (ESS) developed by Marshall Space Flight Center (MSFC). The ESS software is hosted on a VAX computer. It has evolved over the past ten years into a FORTRAN program with 100,000 lines of FORTRAN code. However, it is very time-consuming in using ESS to update the crew activity timeline for the SLS missions. A joint effort between MSFC and Jet Propulsion Laboratory (JPL) of NASA went on for four years to develop an AI-based companion interactive graphic timeline editor, called PLAN-IT (shorthand for Plan-Integrated Timelines). PLAN-IT is a frame-based functional timeline manager. The objective was to enable the timeline engineers to explore scheduling options, recognize scheduling opportunities and thereby include additional or better-arranged activity into a schedule. The origin of PLAN-IT can be traced back to the AI planner DEVISER system of Vere (Ref. 13). Unfortunately, this joint effort of MSFC and JPL was terminated in October, 1988, and the PLAN-IT scheduling system was left unused in MSFC since then. Three main reasons for not using PLAN-IT were: (1) the integration of PLAN-IT into the ESS was poor, (2) the response time of PLAN-IT was unacceptable in certain cases, and (3) the resistance from the ESS developer and user communities was strong.

MIO of GE Government Services obtained the original PLAN-IT source code from JPL in 1988. We have enhanced the software for use in the scheduling of Spacelab experiments to support the Spacelab Life Science missions. The enhanced software SLS-PLAN-IT Scheduling System was used to support the real time reactive scheduling task during the SLS-1 mission. This software will be further enhanced for the SLS-2 mission, and is expected to completely replace the ESS Flight Planning System (ESS/FPS) currently in use by the MIO in the SLS-3 time frame. The SLS-PLAN-IT is currently hosted on the TI Micro-Explorer Lisp machine and will be ported to the SUN workstation under the Common Lisp Object System (CLOS) environment.

The project objective of SLS-PLAN-IT is to provide an intelligent scheduling tool that will allow the timeline engineers of the Payload Activity Planning team to interactively update an ESS generated timeline in a way that is time and cost effective. SLS-PLAN-IT is a decision support tool. Its purpose is to aid an expert human scheduler not only with effective graphics and a menu-driven interface, but also with natural problem presentation.

A vital feature of SLS-PLAN-IT is its resource timelines, which are similar to timelines normally in use. The timeline display shows the scheduler the conflicts in a trial sequence so

that the sequence can be modified and improved immediately. The sequence can be modified and the strategy can be directed while a strategy is running. There are several advantages to this approach. It allows the user to understand easily what is happening. The trial sequence is displayed directly on the screen. As the sequence changes incrementally, the user can quickly grasp what is happening and interact with the scheduling process. He can focus on some aspect of scheduling without being concerned of the other constraints. This feature also makes it easier for the user to capture expert advice.

There also exist controls that allow the user to focus on a strategy. For example, the user can disable some of the resource timelines so that the strategies will only consider a subset of the resources. After the basic schedule is laid out, additional resources can be evaluated. Another control approach is to tell SLS-PLAN-IT to work with certain types of activities or to consider moving activities within a user-defined window. The effect of SLS-PLAN-IT's strategies can be reduced by freezing some activities since only the user can move frozen activities. One of the more effective control approaches allows the user to select a single activity to which a strategy can be applied so that he can ask SLS-PLAN-IT if there is a better place in the sequence for this activity. This feature provides the scheduler with a "smart" sequence editor.

The goal of SLS-PLAN-IT is to achieve a blend of human and machine expertise. SLS-PLAN-IT initially produces preliminary layouts. After political decisions have been made, they will be reflected in the schedule. The operator can direct SLS-PLAN-IT to make minor changes in the sequence, or he can control the strategies. Finally, the operator can use SLS-PLAN-IT as an editor to verify that certain constraints have not been violated.

In the following sections, we will first describe the problem domain, review the relevant literature, then give a detailed description of the system architecture, report the current status and enhancement plan of the project, and finally discuss what we have learned from the project.

The Problem Domain:

Mission planning schedules are composed of three types of element: activities, resources, and constraints. Activities are the events in a schedule. They can either have durations (like experiment steps) or be point events (like a space shuttle launch). Activities consume, create, or replenish resources. Activities also have inter-relationships that are often expressed as precedence relationships or concurrency/non-concurrency of activities. Resources can be associated with one activity, a group of activities, or all activities. There are activity-specific resources, e.g., equipment associated with an experiment, and pool-resources, e.g., electrical power.

The Spacelab mission planning is an overconstrained domain.

In past Spacelab or Skylab missions, low priority experiments were occasionally bumped to achieve more important goals. Therefore, the mission planners must be able to relax or even ignore certain constraints in order to get an acceptable schedule.

In our timeline engineers' terminology, a performance is an execution of an experiment, and a step is an activity of an experiment. The experiments are then modeled by the constraints of the steps involved and the constraints of the performances. The constraints imposed for the Spacelab missions can be categorized into time constraints and resource constraints. The time constraints include performance time window, maximum and minimum performance duration, maximum and minimum performance delay, maximum and minimum step duration, maximum and minimum step delay, concurrency and non-concurrency of steps, and target or attitude opportunities. The resource constraints include equipment, nondepletable resources, depletable resources, resource carry-through, crew selection, crew lock-in, crew monitoring, and the requirements of balanced resource usage. In fact, this is an NP-hard scheduling problem.

Literature Review:

Bennington and McGinnis gave a survey of the past research in resource constrained project scheduling problems (Ref. 1). They demonstrated how to search for the optimal algorithm by three basic approaches: the first approach was to formulate the problem as an integer linear programming (ILP) problem, which can be solved by standard ILP techniques; the second approach was to directly employ some enumerative scheme for constructing an optimal schedule; and the third was to formulate the problem in terms of minimaximal paths in a disjunctive graph, which could be solved by network flow methods or implicit enumerations.

In spite of the progress in research, almost all researchers have agreed that the heuristic method is still the only viable solution technique for large-scale practical problems since the computing time would be prohibitively large if exact optimal procedures were used. Studies of the complexity of the resource constrained scheduling problems also draw lots of attention. Coffman showed that these problems were actually NP-hard (Ref. 3). Elmaghraby (Ref. 5) and Coffman (Ref. 3) contain excellent coverage of the recent results in resource constrained scheduling problems.

In recent years, the emergence of expert system technology has had a great impact on scheduling system design. Dhar and Ranganathan used the university course timetable scheduling problem as an example to contrast the advantages and disadvantages of AI approaches versus OR approaches (Ref. 4). They pointed out that the OR approaches had the following disadvantages:

1. Single objective limitations: The objective function used in OR formulation express one goal, but there are other goals

that the scheduling expert tries to satisfy.
2. Compiled knowledge limitations: Solutions are very sensitive to the coefficients of the objective function, and some default knowledge is difficult to incorporate into the coefficients.
3. Global optimization limitations: Global optimization essentially obscures the reasons for assignments and implies lack of explanation for its decisions.
4. Lack of support in making plan revisions: Plan revisions are inevitable, but it is very difficult for the decision maker to revise the schedule with minimum perturbation in OR approaches.

Jaap and Davis described an interesting review of the software development of ESS (Ref. 8). The ESS software hard-coded the scheduling rules in FORTRAN to handle the time constrains and the resource constraints. The scheduling core of ESS consisted of five modules: the bookkeeper for resource tracking, the checker for determining availability of resources, the loader to load the schedule, the trace listing as an explainer, and finally the selector to determine the ordering of scheduling the activities. Two methods, the random-order method and the preference-order method, were incorporated in ESS.

Boarnet documented the requirements of a scheduling expert system tool from NASA's point of view (Ref. 2). It was one of the best examples of the impact of expert system technology on the design of the scheduling system software. In the paper, Boarnet discussed the requirements of a scheduling expert system tool for Space Station Freedom mission planning applications. He pointed out that the scheduling tool should represent activities, resources, and constraints, with facilities to group those elements and to represent the time variance of the elements. The tool should support activity scheduling and job scheduling. Enumeration of alternatives with algorithms, hypothetical worlds, rule systems, and schedule hierarchies should be integrated into a powerful reasoning tool. The tool must support the procedural code that might be necessary either for procedure attachment or to control the scheduling techniques. The tool must support interactive scheduling with intelligence that can be interactive or automatic at the user's discretion, and with good human factors.

In the panel discussion on "AI-Based Schedulers in Manufacturing Practice" held in IJCAI-1989, Detroit, USA, Sidhu (Ref. 10) pointed out that the most common mistakes in building intelligent scheduling system include:
1. Inadequate analysis of dominant domain characteristics, especially when prepackaged scheduling tools are used.
2. Inappropriate reliance on locally greedy strategies. Because most scheduling problems are fairly complex, they are often simplified by using simple local dispatching rules.
3. Misuse of shallow expert knowledge: Human schedulers always over-simplify the constraints, or misrepresent situation-dependent knowledge as general-purpose knowledge.

The special issue of AI magazine, January 1991, contains a report of the workshop, "Issues in the Design of AI-Based Schedulers", by Kempf et al. (Ref. 9). The issues covered in the workshop included expert vs. deep vs. interactive schedulers, integrating predictive and reactive decision-making, maintaining convenient schedule descriptions, and some other advanced topics like learning and benchmarks. Several points expressed by the participants are very interesting and representative:

1. Fully automated schedulers are not as desirable as interactive schedulers because the man and the machine bring complementary skills to the scheduling task.
2. Many deployed scheduling systems contain only a small amount of AI. Successful systems can be dominated by other issues such as the user interface, database connections, and real-time data collection.
3. One strong point for interactive methods is that they allow humans to build schedules by methods that they naturally use but are hard to represent, and allow humans to guide the search.
4. Integration of predictive and reactive scheduling components is important. A blackboard-style scheduling system architecture may be appropriate.
5. Optimization is an ill-conceived objective for scheduling. It is hard to define, and factory operations are unpredictable.

Fox and Smith proposed a knowledge-based system for factory scheduling called ISIS (Ref. 6). The central idea of ISIS is that schedule construction can be cast as a constraint-directed activity that is influenced by all relevant scheduling knowledge. In the paper, they pointed out that given the conflicting nature of the domain's constraints, the problem differs from typical constraint satisfaction problems, and one cannot rely solely on propagation techniques to arrive at an acceptable solution. Rather, constraints must be selectively relaxed and the problem-solving strategy must be one that finds a solution that best satisfies the constraints. This implies that the constraints must serve to discriminate among alternative hypotheses as well as to restrict the number of hypotheses generated. The design of ISIS focused on two issues:

1. Construction of knowledge representation that captures the requisite knowledge of the job shop environment and its constraints to support constraint-directed search, and
2. Development of a search architecture capable of exploiting this constraint knowledge to effectively control the combinatorics of the underlying search space.

In constructing a job shop schedule, ISIS conducts a hierarchical multi-level constraint-directed search in the space of all possible schedules. The different levels of the search provide multiple abstractions of the scheduling problem, each a function of the specific types of constraints that are considered at that level. Control generally flows in a top down fashion, and communication between levels is accomplished via the exchange of constraints.

Syswerda and Palmucci presented the construction of a genetic algorithm based optimizer for a resource scheduling application (Ref. 12). Genetic algorithms (GA) use Darwin's fitness-for-survival principle to do function optimization. The optimizer described in the paper is a combination of local expert search and global search provided by a genetic algorithm. The issues involved in the construction of a GA-based scheduler include:

1. how to represent the schedule as a bit-string used in GA,
2. how to isolate the details of the problem from the GA.

They also pointed out that the system must be able to combine manual scheduling of special cases with automatic scheduling based on more general criteria. Manual scheduling is accomplished by the use of an intelligent graphical interface. The interface is intelligent in that it understands all the well-defined constraints of the scheduling problem, and advises the user about where to place tasks while disallowing the construction of illegal schedules. We have similar graphical user interface in SLS-PLAN-IT.

SLS-PLAN-IT's System Architecture:

SLS-PLAN-IT's approach to problem solving relies on three highly interactive elements: a model builder to construct activity and resource models, a user interface that takes into consideration what the user needs to know and how he controls or directs the scheduling process, and the scheduling strategies.

Hayes-Roth (Ref. 7) used a blackboard model to implement their opportunistic strategy, planning both top-down and bottom-up. Smith et al (Ref. 11) reported an extension of ISIS to OPIS (the Opportunistic Intelligent Scheduler), which was implemented with a blackboard style architecture. These knowledge sources had implemented alternate scheduling strategies that extended and revised a global set of scheduling hypotheses. Smith et al. reported better performance than that of ISIS with the multiperspective scheduling approach. These blackboard models have beed adopted by SLS-PLAN-IT scheduling system.

The blackboard structure is a global, hierarchical data structure partitioned to represent the problem domain as a hierarchy of analysis levels. Each level consists of nodes that are objects in the system implemented as frame structures. The nodes are integrated by links, where a node in the hierarchical structure represents an aggregation of lower level nodes. Thus, the blackboard can be structured as an undirected graph of nodes. However, one can place nodes without links on the blackboard. During problem solving, partial schedules begin to grow on the blackboard. The higher levels represent abstract decisions made about the general pattern of the mission schedule, while the lower levels represent decisions made about the specific details of the schedule. The relationships of the nodes are either specified by the model builder prior to the scheduling sessions, or specified via the mouse by the user dynamically during the

manual-mode scheduling sessions. Thus, knowledge sources can create decisions that refine the schedule from the higher to the lower levels of the blackboard, growing the schedule in a top-down fashion. Alternatively, knowledge sources can create decisions about specific details of a schedule and incorporate those decisions into the whole schedule, growing the schedule in a bottom-up fashion. The knowledge sources are specialists that access the blackboard by creating nodes, modifying nodes, or modifying links between nodes. This allows a knowledge source to contribute information without knowing which other knowledge sources will be using the information. In SLS-PLAN-IT, the knowledge sources are implemented as scheduling strategies that can be triggered whenever a goal is posted or whenever data changes. The three main components of the system are described in detail in the following sections.

Model Builder

SLS-PLAN-IT uses a datatype specification modeling language to model the scheduling requirements of the mission. The purpose of having the modeling language is two-fold. Firstly, it is to simplify resource definitions programming so that classes of items already defined need not be re-coded by hand. Secondly, it is to insulate the resource-describer from having to know the exact order of resource definition commands that must be included in the source code.

In response to the users' request to remove the major obstacle that discourages the users from using SLS-PLAN-IT, a model builder is currently under development and will be included into SLS-PLAN-IT for the SLS2 mission.

The model builder will be able to construct activity models and resource models. An activity can be an experiment to be scheduled or an electrical storage to be discharged. Resources include the depletables, the non-depletables and the human resource. Examples include the power, the data rate, and the crew.

An activity model will include a series of individual steps to be performed in the experiment, the scheduling time ranges or time allotment, the resources to be used, and the constraints in scheduling. The steps in an activity may occur sequentially or concurrently, or they may overlap one another.

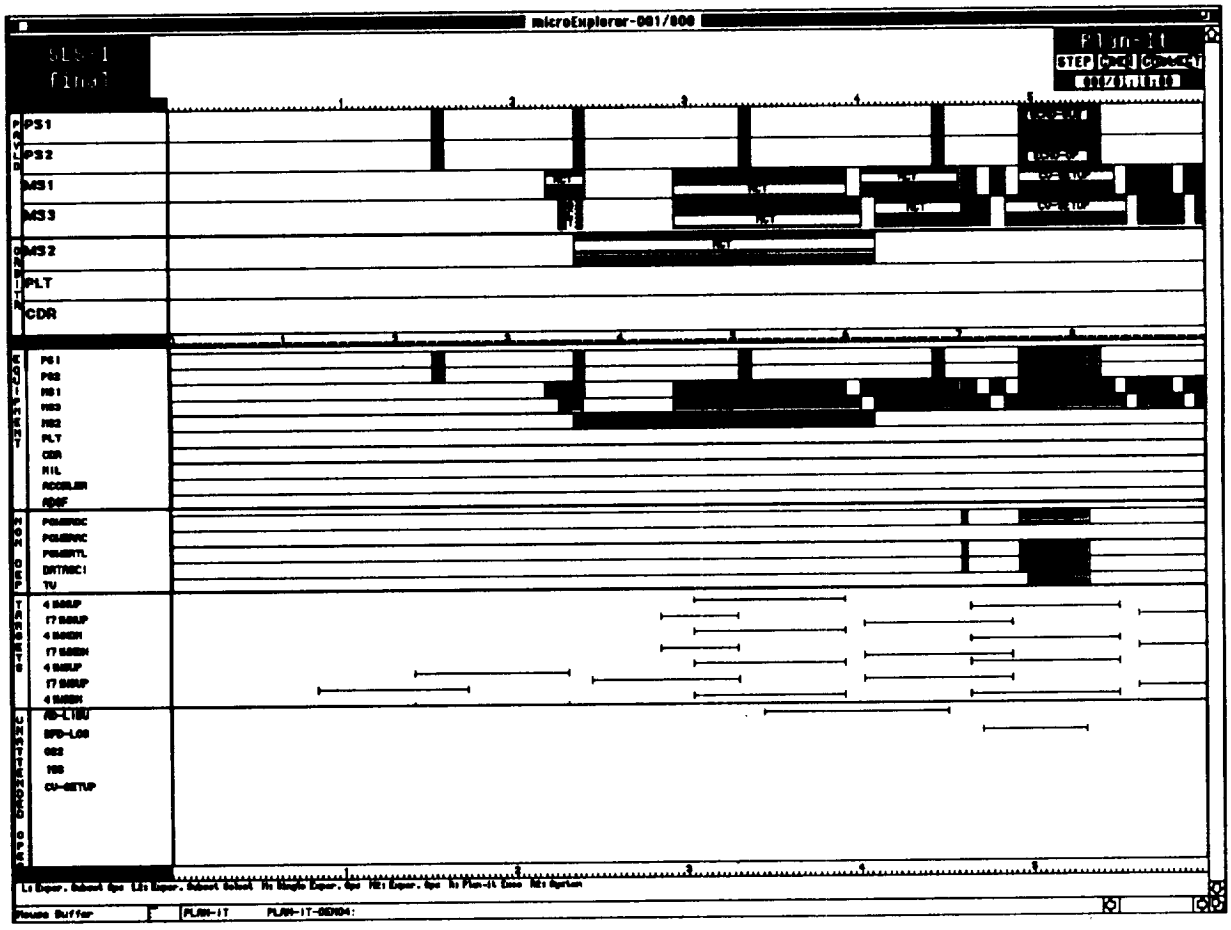
Resources are modeled as timelines that show how each resource is used or changed throughout the entire sequence. A resource model will include the availability of the resource in quantity and time, and the constraints in scheduling.

The activities and the resources interact with each other throughout the scheduling process. Whenever an activity or a step of an activity is changed, the resource timelines will be updated. Whenever the usage of any resource has exceeded its limits, a conflict will be detected. In this way, the resource

models will serve as safeguards against the misallocation of the resources.

User Interface

SLS-PLAN-IT's man-machine interface focuses on the graphical presentation of the resources and activities. Control of the program is through pop-up menus and mouse operations. The screen of SLS-PLAN-IT is divided into six sections (see Figure). The top section includes a status pane that displays operational messages of the program, or the status information of the mouse, or the detailed information related to the timeline interval over which the mouse is positioned. The next five sections are graphical displays of the experiment timelines, the equipment-resource timelines, the non-depletable-resource timelines, the target or attitude opportunity information window, and the unattended-operation timelines, one below another. The unattended-operations consume resources, but no crew were associated with them except for occasional monitoring. The details of an experiment can be edited interactively. The



Figure

resource timelines display white where there is no load, gray where loads exist with no resource conflict, and black to denote a conflicting area. By positioning the mouse over an activity, details of the activity will be displayed in the status pane to aid the user in editing. When the mouse is over a resource timeline, the status pane will display the amount of the resource being used and the activities involved. When the mouse is over a conflicted resource area, the status pane will display the activities that caused the conflict.

When the entire SpaceLab sequence of the mission is displayed, the screen is overwhelmed by the amount of detail. Therefore, SLS-PLAN-IT provides a zooming facility for the user to tailor the screen display to his/her need. The user can examine any portion of the timelines at any specified scale. Since SLS-PLAN-IT's display is interactive, the user can actually watch during the automatic mode what the scheduling strategy is doing as the experiments are being moved and modified. The impact on the resource timelines and on the experiments is directly and immediately shown to the user. At any point in the processing, the user can redirect SLS-PLAN-IT to focus on a different aspect of the sequence.

There are several modes of operation in SLS-PLAN-IT, from running without user interaction to user controlling the search or user manually scheduling the experiments. Therefore, the user can select the level of control over the mission timeline, and go back and forth among the various modes of operation.

An important feature of SLS-PLAN-IT is the explicit conflict representation on resource timeline. This is a natural representation for the expert user and thus made the interaction with the user more direct and simpler. Since the experiments are tracked explicitly on the Gantt-chart type timeline blackboards, the experiments could be scheduled in any random order.

The ability of an expert scheduler to intuitively grasp what the scheduling engine is trying to do is very important, as has been noticed by several researchers as a necessary condition of a successful scheduling system. SLS-PLAN-IT developers are well aware of this.

Scheduling Strategies

Besides the manual scheduling mode supported by the blackboard structure, the constraint propagation mechanism and the graphical user interface, SLS-PLAN-IT also supports automatic scheduling mode with various scheduling strategies. One of the fundamental ideas of SLS-PLAN-IT is that there is no single "correct" way to sequence. In fact, no single way is powerful enough to do the task in a reasonable time. Thus, SLS-PLAN-IT supports a number of scheduling strategies that can then be combined into a scheduling session.

In automatic mode, the system must be able to compare

different partial schedules and choose one to continue scheduling. This requirement is reflected in the structure of scheduling strategy. A strategy consists of three parts. The first part is a goodness measure that indicates whether one sequence is better than another. This measure can change from strategy to strategy. Typically, the goodness measure rates the total conflict on the resource lines. The second part is to select activities to be changed, which can simply be all the activities of certain types or the activities involved in the worst conflict. The third part is to suggest the actions to be taken such as to move, to modify, or to delete an activity. SLS-PLAN-IT makes small changes, one at a time, to improve a goodness rating.

Several strategies are currently implemented in SLS-PLAN-IT. These strategies together form a hill climber. A goodness rating will determine a topology for the search space. A strategy will change the schedule until it finds a local maximum in the topology of the search space. By selecting a different strategy, the topology of the space will be changed and the SLS-PLAN-IT will be able to continue improving the mission sequence.

SLS-PLAN-IT possesses meta-knowledge in the form of strategy modifiers. These modifiers restrict the search space of a strategy. An example of this is the restriction on the number of resources a strategy could consider. This particular modifier is based on the knowledge that, to the first order, a mission schedule is determined by a small subset of the total number of resources. Other modifiers force a strategy to only consider moving experiments to areas of the mission timeline that have little resource usage.

There seems to be no single best way in scheduling. The scheduling techniques depend on the particular project, the life point in the mission, and the current schedule. SLS-PLAN-IT is able to represent many different scheduling strategies. Flexibility in choosing a suitable scheduling strategy is the key to successful scheduling. The concept of scheduling strategy provides a natural hook of SLS-PLAN-IT system to any optimization technique. Given a goodness measure as the objective function, all the scheduling and sequencing techniques available from traditional operations research discipline or non-traditional combinatorial optimization approaches can be incorporated into SLS-PLAN-IT in the form of scheduling strategies.

Current Status of SLS-PLAN-IT:

MIO of GE Government Services obtained the original PLAN-IT source code from JPL in 1988. We have tailored the software for use in the scheduling of Spacelab experiments to support the Spacelab Life Science missions. Although the original PLAN-IT has the ability to perform very specialized strategies to resolve particular scheduling difficulties, the automatic mode that uses the above strategies is still not powerful enough to handle the overconstrained resource requirements of scheduling the Spacelab

mission timeline. MIO's current major concerns in SLS-PLAN-IT to support the SLS-2 mission include the graphical user interface and the automatic constraint propagation capability, which allow the user to modify a timeline by mouse operations. We will enhance the automatic scheduling capabilities for SLS-3.

A Spacelab mission timeline contains over fifty resources and hundreds of experiments. The timeline engineers can manually enter the initial schedule or use MSFC's tabular-form scheduler ESP to produce the initial schedule. During SLS-1 mission planning phase, the timeline engineers used SLS-PLAN-IT to maintain the schedule produced by ESP. After the schedule was modified by SLS-PLAN-IT, the modified schedule was transmitted to the Flight Planning System (FPS) for standard output plotting.

The SLS-PLAN-IT Scheduling System was used to support the Spacelab Life Sciences-1 (SLS-1) mission during the mission period from 6/05/91 to 6/14/91. This on-line real time usage of SLS-PLAN-IT during the mission demonstrated the strength of this scheduling system. The timeline engineers of the Payload Activity Planning (PAP) team confirmed that SLS-PLAN-IT is a flexible and useful scheduling tool that provides a real time reactive planning capability that the old scheduling system ESP/FPS lacks. For example, the timeline engineer had used SLS-PLAN-IT to reschedule the activities on flight day 9 due to short notice. The ESP required more time than available to do this kind of replanning. The SLS-PLAN-IT had won the user confidence and acceptance that were not in existence in the early stage of PLAN-IT development.

Feedbacks from SLS-PLAN-IT users during SLS-1 mission are:

1. The system gives visual display of experiment timeline with schedule conflicts indicated. Mission planning using SLS-PLAN-IT is much quicker than using ESS.
2. The mouse and menu-driven user-interface of SLS-PLAN-IT and the Gantt-chart like resource timeline are very convenient to support manual-mode scheduling of the mission. Minimum user training is needed for manual-mode scheduling if SLS-PLAN-IT is used, instead of the six months training time of ESS.
3. It is usable as a real-time reactive mission scheduler with prospects of increasing productivity of mission support staff and increasing science returns of the Spacelab experiments.
4. A quicker and more convenient model-builder is needed to support the SLS-2 mission. The integration of SLS-PLAN-IT with other flight planning software needs to be improved.

Enhancement:

The on-line real time usage of SLS-PLAN-IT aroused the user interests to further enhance the SLS-PLAN-IT software. The major enhancement requirements include:

1. rehosting SLS-PLAN-IT to a SUN platform to boost the operation speed and to allow better integration,
2. providing an intelligent model builder to enhance the model editing capability and shorten the modeling and planning time,

3. providing more options for automatic file creation and generation of operation output in current FPS format, which includes the information of ground tracking, attitude timeline, sun/shadow times and all other miscellaneous information on other FPS output.
4. additional capabilities to support the execute shift activities; the exact requirements are to be determined.

The direction we are taking is to completely replace the FPS system with SLS-PLAN-IT in the SLS-3 time period. With a very high level of user involvement, the SLS-PLAN-IT will evolve as a fully automated knowledge-based scheduling system with graphical user interface for space exploration.

In summary, the performance of SLS-PLAN-IT during the SLS-1 mission was very satisfactory. Recommendation for further enhancements of the software was made for the SLS-2 mission. It is expected that the SLS-PLAN-IT will completely replace the ESS currently in use by the MIO in the SLS-3 time frame.

Conclusion and lessons learned:

During the development of SLS-PLAN-IT, we have gained some useful experience in software engineering of an AI-based scheduling system that we would like to share with the community:

1. Quick response time is crucial in real time scheduling environment. One of the reasons that JPL's version of PLAN-IT was abandoned is that it did not have a model editor. It took an hour or more on Symbolics 3650 to incorporate the model from ESS. We improved the operation time to about ten minutes in the first version of SLS-PLAN-IT. Enhancements of the model editor to support incremental model editing are in progress.

2. Good integration of the AI scheduler with all other Flight Planning System (FPS) software is important because the purpose of SLS-PLAN-IT is for operational daily usage to support the mission.

3. Automatic shift of focus is difficult to achieve. There are many scheduling strategies available in the automatic mode of PLAN-IT. However, the users did not use them for the SLS-1 mission. One of the reasons is that the users do not fully comprehend the scheduling strategies. We have to better express the strategies to the users in more natural ways or the "automatic mode" will stay unused.

4. It is very important to allow the users to play "what-if" games during scheduling process and see why things happened. This is one of the reasons the MIO mission planners switched from using ESS to using SLS-PLAN-IT.

5. User-naturalness is the key to have a good user interface. For example, the automatic constraint propagation capability of SLS-PLAN-IT and the blackboard structure of the resource lines

are user-natural tools to support the above vital features.

6. The level of user involvement and expectation of SLS-PLAN-IT is very high in MIO over MSFC and JPL. In fact, the users always expect more than the developers can provide. We are driven by the users and the users are driven by the scheduling workloads they support.

7. The effects of a schedule change should be kept as local and as minor as possible. Minimum disruption of the schedule is sometimes more important than obtaining an optimal schedule.

Our experience with SLS-PLAN-IT reconfirms the observations made in the IJCAI workshop mentioned earlier. Fully automated schedulers are not as desirable as interactive schedulers because the man and the machine bring complementary skills to the scheduling task. Also, deployed scheduling systems contain only a small amount of AI. The issues of user interface, database connection, and real-time requirements dominate the system design and user acceptance of the scheduling system. The most important feature emphasized in SLS-PLAN-IT is the user-natural interface to cooperate with humans in changing perspective or focus level to support the opportunistic scheduling strategies. The various strategies employed in the automatic scheduler are attempts to simulate the opportunistic scheduling capability of the human.

Acknowledgment:

The authors would like to thank Mr. Michael Hollander and Mr. William C. Eggemeyer of Jet Propulsion Laboratory for providing us the source code of the original PLAN-IT software and giving us valuable advice during the software conversion period. We would also like to thank all the timeline engineers of MIO/GEGS, Houston, Texas, for giving us their requirements and user feedback concerning the SLS-PLAN-IT Scheduling System.

This work is performed by GE Government Services, Johnson Space Center, Houston Texas in support of the NASA Mission Management Office government contract NAS9-17884.

REFERENCE:

1. Bennington, G.E., & McGinnis, L.F. (1972). A Critique of Project Planning with Constrained Resources. Symposium on the Theory of Scheduling and its Application, Springer-Verlag, New York, 1973.

2. Boarnet, M.G. (1986). Requirements for a Scheduling Expert System Tool. NASA/JSC Mission Planning and Analysis Division, Internal Memorandum #FM7(86-66), April 1986.

3. Coffman Jr., E.G. (1976). Computer and Job Shop Scheduling Theory, John Wiley and Sons, Inc., New York, 1976.

4. Dhar, V., & Ranganathan, N. (1990). Integer Programming vs.

Expert Systems: An Experimental Comparison, CACM March 1990, pp. 323-336.

5. Elmaghraby, S.E. (1973). Symposium on the Theory of Scheduling and Its Applications, Springer-Verlag, Berlin 1973

6. Fox, M.S., & Smith, S.F. (1984). ISIS--A Knowledge-Based System for Factory Scheduling. Expert System, Vol 1, No. 1, July, 1984.

7. Hayes-Roth, B. (1985). A Blackboard Architecture for Control. Journal of Artificial Intelligence, Vol 26, pp 251-321.

8. Jaap, J., & Davis, E. (1986). Expert Scheduling for Spacelab Mission. Proceeding of Conference on Space Applications of Artificial Intelligence, Huntsville, AL, Nov 13-14, 1986.

9. Kempf, K., Pape, C., Smith, S.F., & Fox, B.R. (1991). Issues in the Design of AI-Based Schedulers: A Workshop Report. AI Magazine, Special Issue Jan. 1991. pp. 37-46.

10. Sidhu, S. (1989). Avoiding Typical Mistakes while Building Intelligent Scheduling System. Panel Discussion on AI-Based Schedulers in Manufacturing Practice, IJCAI-89, Detroit, Michigan, USA, August, 1989.

11. Smith, S.F., Fox, M.S., & Ow, P.S. (1986). Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems. AAAI's AI Magazine, Vol. 7, No. 4, Fall 1986.

12. Syswerda, G., & Palmucci, J. (1991). The Application of Genetic Algorithms to Resource Scheduling. Proceeding of the Fourth International Conference on Genetic Algorithms (ICGA-1991), San Diego, CA, July 13-16, 1991.

13. Vere, (1981). Planning in Time: Windows and Durations for Activities and Goals. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 5, No. 3, pp. 246-267, May, 1981.