# Scheduling Lessons Learned from the Autonomous Power System

Mark J. Ringer
Sverdrup Technology Inc.
NASA Lewis Research Center Group
Cleveland, Ohio 44135
Ringer@mars.lerc.nasa.gov

## Abstract

The Autonomous Power System (APS) project at the NASA Lewis Research Center is designed to demonstrate the applications of integrated intelligent diagnosis, control and scheduling techniques to space power distribution systems. The project consists of three elements: the Autonomous Power Expert System (APEX) for Fault Diagnosis, Isolation, and Recovery (FDIR); the Autonomous Intelligent Power Scheduler (AIPS) to efficiently assign activities start times and resources; and power hardware (Brassboard) to emulate a space-based power system.

The AIPS scheduler has been tested within the APS system. This scheduler is able to efficiently assign available power to the requesting activities and share this information with other software agents within the APS system in order to implement the generated schedule. The AIPS scheduler is also able to cooperatively recover from fault situations by rescheduling the affected loads on the Brassboard in conjunction with the APEX FDIR system.

AIPS served as a learning tool and an initial scheduling testbed for the integration of FDIR and automated scheduling systems. Many lessons were learned from the AIPS scheduler and are now being integrated into a new scheduler called SCRAP (Scheduler for Continuous Resource Allocation and Planning). This paper will serve three purposes: an overview of the AIPS implementation, lessons learned from the AIPS scheduler, and a brief section on how these lessons are being applied to the new SCRAP scheduler.

## 1. Introduction and Motivation

Future NASA spacecraft and planetary surface installations will require larger and more sophisticated infrastructure systems and living environments. Such systems will consist of dozens of resources and hundreds of attached loads. The electrical power system on the Space Station Freedom, a Lunar base, or Martian base represents a critical portion of such a system. The APS project explores intelligent hardware and software architectures for efficient system operation and scheduling of an electrical power system [Ringer 1991].

### 1.1 The Need For (Automated) Scheduling

Onboard a complex spacecraft many activities must be performed, each competing for a multitude of temporal positions and limited resources. A scheduler must assign start times to each activity without violating any resource or temporal constraints. The resources onboard such a spacecraft will be vastly oversubscribed, having many times more resource requests than available resources. This makes it a paramount objective to efficiently utilize the available resources in order to complete as many activities as possible.

Current NASA space-based systems rely on ground-based human-intensive scheduling methods. Humans provide the main scheduling intelligence for constructing schedules. These schedules are then transmitted to the spacecraft to be executed. If the scheduling expertise and computers are ground-based, every anomaly that occurs onboard the spacecraft that incurs a schedule modification would cause significant time delays and efficiency losses. With the advent of more complex space-based systems such as the Space Station Freedom and beyond, a more efficient automated scheduling paradigm is necessary [Britt 1988].

### 1.2 The APS Project Scheduling Goals

The goal of the APS project is automated scheduling for space systems with proof-of-concept demonstrations on a power system testbed. In this process only the high level goals of the system are stated by the human operators, that is, which activities should be performed. This information is taken and the scheduler attains the goal of activities executed. The scheduler must

not only know how to generate the schedule, but must also know how to implement the schedule, and how to recover from system or load induced deviations in the schedule.

## 2. AIPS Implementation

Since scheduling cannot take place in a vacuum, the scheduler must be able to interact with other agents as well as cope with many operational concerns. The scheduler must be able to generate an initial schedule, it must have domain specific knowledge of how to implement the schedule, it must be able to reactively modify the schedule in the case that the assumed information of the state of the system changes, and it must be able to do this within metric time constraints.

### 2.1 What is being scheduled

The APS Brassboard is a power system testbed that contains a set of power supplies, switchgear, and loads that emulate a space-based power system. This hardware is controlled by a set of embedded controllers capable of configuring the state of the Brassboard. These controllers are then used to configure the Brassboard to supply power to the loads designated by the scheduler. Figure 1 shows the current configuration of the APS Brassboard. RBI's and RPC's are remote controlled switches and an *L* represents a load attached to the system.

The loads attached to the Brassboard are resistive load banks. In order to more closely emulate a space-based power system each load is given a set of attributes resembling those of a space-based system. Each activity (load) has a time varying profile of power demand, earliest start time and latest completion time constraints, priority, and temporal placement preference.

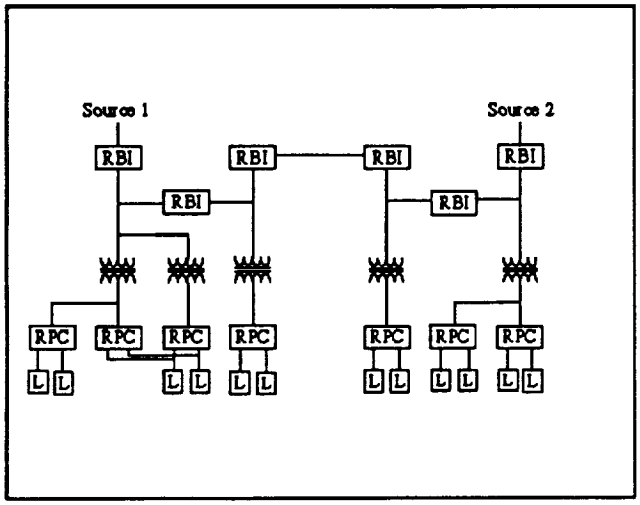


**Figure 1** Brassboard Power System Configuration

### 2.2 Cooperation Between AIPS and APEX

AIPS is responsible for assigning the power requesting activities attached to the Brassboard temporal positions and resources without overallocating the available power. APEX is responsible for the implementation of the schedule generated by AIPS. In order to adequately model the interaction between APEX and AIPS, a set of protocols was developed to communicate different scheduling and rescheduling procedures. Protocols were developed to generate an initial schedule and modify executing schedules. Figure 2 shows a graphical representation of a schedule generated by AIPS. A chart showing the interaction between the three portions of the APS project is given in Figure 3.

### 2.3 Scheduling Methods Used

Two modes of schedule generation are needed for any integrated scheduling system. The ability to generate an initial schedule and the ability to modify (reschedule) an already executing schedule in the case of an anomaly. In the former case, a metric amount of time is allocated to the scheduler after which a solution must be returned. In the latter case, the rescheduling results are usually needed as soon as possible.

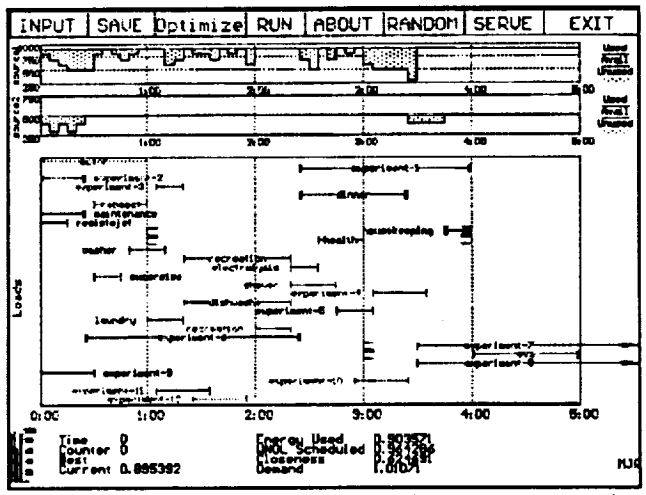


**Figure 2** Representative Schedule Generated by AIPS

The AIPS scheduler has two modes of schedule generation used for scheduling and rescheduling. The scheduling engine is an incremental scheduler that uses a set of activity selection and placement heuristics [Sadeh 1989]. These heuristics are used to construct a schedule by taking each activity one by one, and determining where to place the activity on the timeline. These heuristics also form the basis of the rescheduling engine.

When the scheduler is given more time, it will use the same basic heuristics along with a Monte-Carlo type optimization method to generate multiple schedules

based on the heuristics. Since the heuristics use local goodness information, they do not produce globally good schedules. Small perturbations to these heuristic decisions will often improve the efficiency of the generated schedule. Each schedule is rated based on a goodness rating and when time to generate a schedule has run out, the best schedule (that has been saved in memory) is returned. With a relatively huge state space of solutions this method works quite well probing many portions of the state space that look promising based on the heuristics.

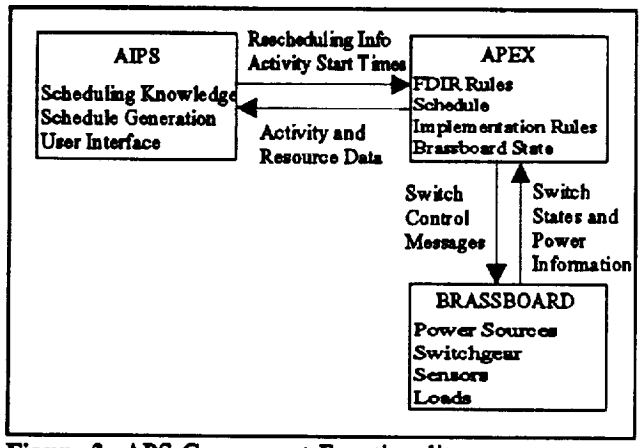


**Figure 3** APS Component Functionality

Rescheduling must also be accomplished "non-nervously", that is, with as little deviation to the original schedule as possible [Biefeld 1990, Zweben 1990]. In systems with human interaction the original schedule should be followed as closely as possible in order to not disturb the humans interacting with the system. To accomplish non-nervous rescheduling, AIPS uses a set of heuristics that judge the amount of perturbation caused by a schedule modification versus the change of goodness of the new schedule.

## 3. Lessons Learned

Lessons were learned from the design of the scheduler, implementing a scheduler in a real system, and integrating scheduling with an FDIR system. Some of the lessons learned represented shortfalls in the original AIPS scheduler while others represented ideas for the improvement of the overall efficiency of the scheduling system.

### 3.1 Retrospective

Many of the concepts implemented in the AIPS scheduler worked quite well. Time was broken down into smaller scheduling horizons in order to make the problem solution feasible. Priorities were used to delineate between the relative value of activities. Time was partitioned at a granularity of five minutes. This was a reasonable simplification since the time for APEX and the Brassboard to be configured was on the order of one minute. The ability to schedule within metric time constraints was incorporated. A graphical interface was available for both schedule display and human-scheduler interaction.

The largest assumption made about the environment was that all temporal durations and resource requests are exact. In a real-life situation, if an activity requests 100 watts for one hour the probability of the activity using a constant 100 watts or lasting exactly one hour is quite small. The problems incurred may include undervoltage/overcurrent conditions caused by higher than expected demands as well as propagation of temporal constraints among activities caused by an extension of an activity's duration. The need for some type of temporal or resource padding is necessary. This padding decreases schedule efficiency although it may improve overall implemented schedule efficiency since the schedule will not have to be modified as often with the padding added.

### 3.2 Perspective

Much was learned about scheduling, but even more was learned about implementing a schedule in an automated domain. The whole object of scheduling is to produce the best overall system efficiency. In order to increase the efficiency of the implemented schedule, most new ideas point to the need for the ability for real-time reaction in the scheduler [Johnston 1989]. Here are three examples.

Conventional schedulers use temporal padding to increase the probability of executing a schedule. Temporal uncertainties cause the forward propagation of predecessor/successor constraints and resource availability. If activities are padded, and this padding is not used, it is wasted. It may be possible, however, to assign this temporal position/resource to another activity. This would entail moving another activity forward in time to fill the temporal position/resource left unused by the previous activity. This demonstrates a need for reaction in the scheduler.

Suppose a 500 watt cooling fan operates only when the experiment temperature rises above a certain threshold. This may only operate 10% of the time (10% duty cycle). How can the resources be allocated to prevent oversubscriptions? If 500 watts are continuously allocated 90% of this energy will be wasted (of course, a conflict free schedule is guaranteed). Energy balancing between multiple duty cycle activities can be used, but problems arise if all these activities turn on at the same time. Reaction is needed to delay some of these events if they desire to consume power when it is not immediately available. In addition, there is the possibility of

performing energy balancing in the power system domain. With energy balancing however, it is necessary to use a storage type resource such as a battery.

When using reaction, think about the "moveability" of an activity. In a Space Station domain, a dishwashing activity is much more moveable than a medical experiment using two crew members and various ground-based experts. The dishwashing activity has very few attached dependencies while the medical experiment would require the movement of many human interactors. The dishwashing activity is easier to move and a small temporal position change will not affect it as long as the dishes are washed before the next meal. This information can be used to make reactive modifications to the schedule without impacting the humans who will have to interact with the system.

The ideas of temporal padding usage, duty cycle balancing, and activity moveability will allow for more efficient use of limited resources. Of course, a scheduler (and testbed architecture) that allows these ideas to be implemented remains to be built and tested. The next section will briefly describe this new scheduler.

## 4. Implementing the Lessons Learned

The SCRAP scheduler is currently under development. General improvements in the representation of the SCRAP scheduler include multiple resources, multiple resource types (capacity, consumable, and storage), one second time granularity, activities broken into tasks, and multiple levels of schedule abstraction. Since the previous section showed a need for reaction, a scheduling paradigm that makes reaction easier would be beneficial.

### 4.1 Prediction vs. Reaction

Two general categories can be delineated in scheduling: predictive and reactive systems. Predictive scheduling allows the efficient allocation of available resources to activities by generating schedules based on predicted knowledge of the activity and resource states. This type of scheduling works well in static domains but is often hard to implement and less efficient in complex, uncertain, and dynamic domains. Reaction provides easier implementation in dynamic domains, but sacrifices resource usage efficiency caused by the lack of knowledge used to generate schedules.

In most real world problems a combination of static and dynamic domains exist. For example, a completely reactive scheduler might have no information on predicted resource demands of an activity, while a completely predictive scheduler would assume exact temporal durations and resource requests. Usually, a combination of these methods are used with a predicted resource level that provides an allowance for deviations from that level. This would point to the use of a combination of reaction and prediction. All schedulers that operate in a real domain actually combine the two, but the idea of SCRAP is to provide a framework that allows these ideas to be implemented efficiently.

### 4.2 How to Combine Prediction and Reaction

Even though building an initial schedule is computationally intense, the need to continuously modify the schedule during execution is even more difficult because of the tighter time constraints in the rescheduling domain. When rescheduling, all temporal and resource constraints propagate forward causing even more conflicts in the schedule, also known as the *ripple effect.* Propagating temporal and resource constraints during a reschedule clobbers previously computed future portions of the schedule. If rescheduling occurs often, the entire precomputed schedule may be recomputed by the rescheduling engine. This is an extreme case but proves the point that it may not be necessary to construct the initial schedule with a great level of detail. Therefore it may be wise to schedule far term activities with less effort or detail than near term activities. In the SCRAP scheduler this is accomplished by using multiple levels of abstraction when scheduling activities. Further into the future the schedule is constructed abstractly, while nearer to the execution time more precision is used. Also, more in-depth scheduling methods are used for times nearer to the execution time than for times further into the future.

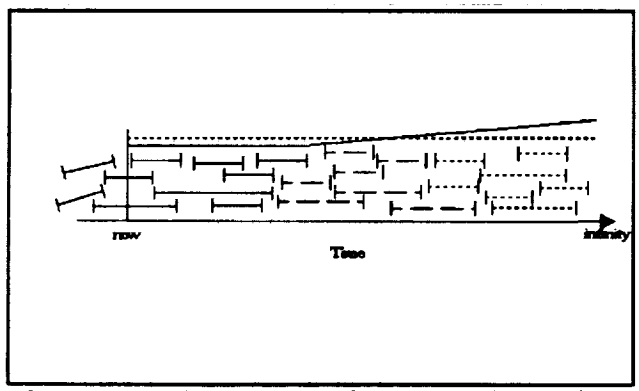


**Figure 4** The SCRAP View of Scheduling

Multiple abstractions based on temporal distance from the execution time will allow for more efficient forward temporal propagation of constraints in the schedule since less information is used for the future portions of the schedule. The future portions of the abstractly generated schedule serve as a partially computed schedule when it comes time to actually schedule at a more precise level. The scheduling timeline can be looked at as a rolling horizon, with the future coming closer to

the present as time ticks during execution.

Figure 4 graphically shows the general idea of SCRAP. The timeline moves in conjunction with the movement of "real" time. Time "now" is the current execution time of the schedule. Time "infinity" is some time very far in the future. The gantt chart shows I-beams at different levels of scheduling abstraction. The solid lines are precisely scheduled, the dashed lines are scheduled at a medium abstraction, while the dotted lines are abstractly scheduled. Resource oversubscriptions are allowed in the future since the schedule in those areas has not been computed more abstractly. Nearer to the time of execution, more scheduling effort and precision is used and these resource conflicts will be eliminated.

Many of the reactive situations stated in the lessons learned section can be more easily implemented using the SCRAP paradigm. In an automated domain the scheduler has much more control over the executing schedule. This control along with the ability to efficiently modify the schedule during execution will allow for an overall implemented schedule efficiency increase.

## 5. Conclusion

The Autonomous Power System project at the NASA Lewis Research Center is an ongoing effort to demonstrate the use of knowledge-based diagnosis and scheduling software in advanced space-based electrical power systems. The APS project has completed one development iteration. A scheduling system was developed for the APS project and integrated with an FDIR system and hardware. The original AIPS scheduler was successful as a learning tool and a new improved scheduler is being developed. Many new ideas for increasing the implemented schedule efficiency will be realized using the SCRAP paradigm. The SCRAP scheduling paradigm will allow for more efficient use of the available resources.

## Acknowledgements

This works was performed under NASA contract NAS3-25266 with Jim Kish as Technical Coordinator.

## References

[Biefeld 1990] Biefeld, E., Cooper, L., "Operations Mission Planner: Final Report", JPL Publication 90-16, March, 1990.

[Britt 1988] Britt, D.L., Gohring, J.R., Geoffrey, A.L., "The Impact of the Utility Power System Concept on Spacecraft Activity Scheduling", Proc. 23rd Intersociety Energy Conversion Engineering Conference, 1988.

[Johnston 1989] Johnston, M., "Knowledge-Based Telescope Scheduling", In Knowledge-Based Systems in Astronomy, Springer-Verlag, 1989.

[Ringer 1991a] Ringer, M.J., Quinn, T.M., and Merolla, T., "Autonomous Power System: Intelligent Diagnosis and Control", Proceedings of the NASA Goddard Conference on Space Applications of Artificial Intelligence, 1991.

[Ringer 1991b] Ringer, M.J., "Autonomous Power System: Integrated Scheduling", Proceedings Space Operations, Applications, and Research Symposium, NASA Johnson Space Flight Center, Houston, Texas, July 1991.

[Sadeh 1989] Sadeh, N., and Fox, M.S., "Focus of Attention in an Activity-Based Scheduler", In Proc. NASA Conference on Space Telerobotics, Pasadena, California, 1989.

[Zweben 1990] Zweben, M., Deale, M., and Eskey M., "Anytime Rescheduling", NASA Ames Artificial Intelligence Branch Technical Report, February 1990.