AUTONOMOUS POWER SYSTEM: INTEGRATED SCHEDULING

Mark J. Ringer
Sverdrup Technology Inc.
NASA Lewis Research Center Group
Brook Park, Ohio 44142

ABSTRACT

The Autonomous Power System (APS) project at NASA Lewis Research Center is designed to demonstrate the abilities of integrated intelligent diagnosis, control and scheduling techniques to space power distribution hardware. 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 determine system configuration, and power hardware (Brassboard) to simulate a space-based power system. Faults can be introduced into the Brassboard and in turn, be diagnosed and corrected by APEX and AIPS.

The Autonomous Intelligent Power Scheduler controls the execution of loads attached to the Brassboard. Each load must be executed in a manner that efficiently utilizes available power and satisfies all load, resource, and temporal constraints. In the case of a fault situation on the Brassboard, AIPS dynamically modifies the existing schedule in order to resume efficient operating conditions.

A database is kept of the power demand, temporal modifiers, priority of each load, and the power level of each source. AIPS uses a set of heuristic rules to assign start times and resources to each load based on load and resource constraints. A simple improvement engine based upon these heuristics is also available to improve the schedule efficiency.

This paper describes the operation of the Autonomous Intelligent Power Scheduler as a single entity, as well as its integration with APEX and the Brassboard. Future plans are discussed for the growth of the Autonomous Intelligent Power Scheduler.

1. INTRODUCTION

Many of the obstacles in the implementation of an on-board integrated scheduling and FDIR system with actual hardware have yet to be investigated. The APS project is meant to research these problems with a set of increasingly complex software and hardware systems. Details of the scheduling aspect of the APS project will be discussed in this paper. A brief graphic of the APS system, each unit's functionality, and communicated information can be seen in Figure 1.

1.1 The Need For Scheduling

Aboard a complex space-based system, many activities must be performed, each competing for a multitude of temporal positions and resources. A scheduler must assign start times to each activity without violating any resource or temporal constraints. Many of the resources aboard 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.

An automated scheduler can be included within a spacecraft, schedule data can be transmitted from a ground base to the spacecraft, or a combination of the two can be employed to accomplish the task of spacecraft scheduling. On-board automated schedulers are still in the research stage and are the topic of this paper. Current Space Station Freedom designs dictate that the schedule be generated on the ground and sent up to the Space Station. All scheduling and rescheduling will be done on the ground [Hagopian 1990]. It may also be possible to have two schedulers, one on the ground, and one in space. The ground-based scheduler will be able

generate very efficient schedules with detailed domain knowledge and large host computers. The on-board scheduler will be able to work on the same problem in "real-time", but do so less effectively because of time and processing constraints.

The problem of scheduling is complex in the areas of computation, domain knowledge, and quantity of data. Computationally, scheduling can be classified as an NP- Hard problem. It is very easy to give a common problem, such as scheduling one day of Space Station experiments, that cannot be implicitly solved in a time comparable to the lifetime of the universe, even on a supercomputer! Building a scheduling engine with enough domain knowledge to be capable of complex reasoning in the area of scheduling is a difficult task. It is also a large bookkeeping task maintaining sufficient information on activity and resource attributes.

1.2 The Need For Automated Scheduling

Future large spacecraft will require larger and more sophisticated infrastructure systems and living environments. Such spacecraft will consist of dozens of resources and hundreds of attached loads. The electrical power system on a platform such as the Space Station Freedom, Lunar base, or Mars base represents a critical portion of such a system. The APS project explores intelligent hardware and software architectures for efficient system operation and scheduling on a representative electrical power system [Ringer 1990, Ringer 1991].

Standard scheduling concerns are beyond the capabilities of humans onboard the spacecraft as well as counter-productive to the science activities planned for the crew of such a vehicle. Ground-based systems are another possibility for generating and implementing schedules. The scheduling computers and humans on the ground would be responsible for schedule generation, while on-board non-scheduling processors would be responsible for the minute-by-minute implementation of the schedule. If the scheduling expertise and computers are kept on the ground, every anomaly that occurs aboard the spacecraft that incurs a schedule change would cause significant time delays and efficiency losses. The relevant information will have to be sent from the spacecraft to the ground, a solution generated, and the information transmitted back to the spacecraft. This will cause a lengthy problem solution time caused by communication and processing delays. This communication delay is compounded for systems such as a Mars base or deep space probes [Dolce 1990].

Since the control of such a large system is difficult to accomplish with the use of ground-based systems, the importance of an automated on-board system is evident. Automated scheduling here means not only automated schedule generation, but also automated schedule implementation. The more responsibilities that an automated system assumes will increase the speed of critical decisions. The use of on-board scheduling for control of such a system will significantly decrease operating costs, increase efficiency, and provide for safer operation [Ringer 1991].

2. AIPS IMPLEMENTATION

The AIPS scheduling problem can easily be broken down into three main areas: schedule representation, schedule generation, and interaction with the other subsystems of APS. Schedule representation is the internal description of the schedule as well as description of the activities and resources modelled in AIPS. Schedule generation describes the scheduling engines used to build the schedules. Interaction describes the interface to APEX and the Brassboard as well as the graphical user interface. Figure 2 shows the main AIPS interface and will be referred to in the next three sections. The remainder of the paper will give an in-depth discussion of each of these three areas.

2.1 Representation

APEX keeps system configuration information, basically representing which loads (activities) are attached to which power sources. Resource availability and activity descriptions are sent to AIPS, which assigns start times to each activity.

The usual method for scheduling continuous-time problems is to break the problem down into multiple subproblems, or planning horizons. These planning horizons represent customary units of time, like one day. Problems of this size are easier to tackle than scheduling all activities for the lifetime of the spacecraft. Once the problem is broken down though, it is necessary to put the pieces back together to create a time-coherent sequence of events. In the AIPS scheduler, it is possible for activities to cross over different planning horizons. It would be unreasonable to assume that at the planning horizon break, all activities would also have a natural break point.

2.1.1 Resources

Capacity type resources are the only type modelled since electric power is the only resource currently available on the Brassboard. Each activity

can only be connected to one power source at a time. There is no representation of power sharing among different power buses, but this case can be represented by combining two sources into one. This connection information is forced by the Brassboard configuration.

Each resource consists of a time-varying profile of available power for the current planning horizon. It is also necessary to include some information about the power availability in the next planning horizon, as discussed in the previous paragraph.

2.1.2 Activities

Each activity is an entity that requests a certain resource (power), from the power sources. (Note: in the scheduling domain this entity is called an "activity", while in the Brassboard domain it is referred to as a "load"). Each activity consists of certain attributes including duration and a list of time variant resource requests. An activity's priority is a relative measure of the activities importance to the user of the activity completing. The activity must also specify to which resource that it is attached. This is actually the job of APEX, which keeps system configuration information. Based upon switch positions within the Brassboard, different paths can be implemented between sources and loads.

Each activity can also have temporal modifiers which specify a time window in which the activity needs to finish. It can also request a general position preference on the timeline, be it early, late, or in the middle of the scheduling horizon or specified time window.

2.2 Generation

Schedule generation is the process of assigning resources and temporal positions to activities. Many types of scheduling engines can be used, with each one having various strengths and weaknesses. Two main factors that can be used to judge a scheduling engine are time to generate a schedule and the schedule's "goodness". These two factors can be compared if the both engines represent the same problem fully.

Some may argue that scheduling and rescheduling are the same problem, with scheduling represented as rescheduling with no activities currently on the timeline. This may be true, but it may still be advantageous to separate the two. In most scheduling domains, a "reasonable" amount of time exists to generate a schedule. In most rescheduling domains, "much less" time is available, any time spent rescheduling the problem is wasted time, i.e. a loss in resource-usage efficiency. For this reason, different scheduling engines are used for the two cases. There is, of course, a tradeoff between the two scheduling methods, faster schedule generation time is traded for less schedule "goodness", but this is made up in the reclamation of the lost time between the anomaly and the "slow" schedule re-implementation (rescheduling).

The rescheduling engine is a heuristic non-backtracking (to re-visit either unscheduled activities, or unchecked time periods) engine that produces a reasonably efficient schedule in a short time. The improvement engine is based on the previously stated engine, but is able to improve the schedule iteratively. Other scheduling engines implement the same concept by using more or less knowledge and/or scheduling methods in the decision-making processes during schedule generation depending on the amount of time allowed [Britt 1990]. Other work in multiple scheduling engines has been focused in constraint-based simulated annealing repair techniques [Zweben 1990a].

The AIPS scheduler and rescheduler use a set of schedule building heuristics based on global knowledge of the resource-based scheduling problem. In this way, a reasonably efficient schedule can be created. The same set of heuristics can also be used in the case that activities need to be removed from the timeline or repositioned on the timeline. The heuristics used in the AIPS scheduling engine can be divided into two categories: those used to select which activity to schedule and those to determine what temporal placement to assign each activity [Sadeh 1989].

2.2.1 Activity selection heuristics

Since the scheduler does not backtrack it is necessary to determine in which order to place activities on the schedule. An ordering of activities is created based on predicted importance to place on the schedule. Once this list is compiled, each activity is then placed on the schedule, if it does not fit, it is ignored and not scheduled. In a non-backtracking engine, an activity is more likely to be scheduled and scheduled in its desired position if it is placed on the schedule as early as possible in the scheduling process. It should be stated that each of the next three classes of heuristics are not used on their own, but a combination of the three is used to generate the ordered list of activities.

> Priority: It is more important to include the higher priority activities on the schedule, therefore, higher priority activities are placed on the schedule first.

Amount of Power Requested: Placing larger resource amount requesting activities on the schedule first usually result in schedules that will consume more of the capacity type resources. If one places smaller resource requests on the schedule first, it is possible that the larger activities will be unable to fit on the schedule. This would result in less capacity type resource usage.

Duration Of Requested Time Window: If an activity has requested that it be placed in a small time window, it is important that this activity be placed on the schedule before others that may take resources within this time window needed by the current activity.

2.2.2 Temporal Placement Heuristics

It should again be stated that the next three classes of heuristics do not stand alone, but all three are used to determine the final position of an activity on the timeline. Each feasible start time for an activity is judged based on the following heuristics. The best of these possible start times is then chosen, and the activity is placed on the schedule.

Projected Resource Demand: This is a conflict avoidance strategy, as opposed to a reactive conflict resolution strategy [McLean 1990]. Bottleneck areas (areas of resource oversubscription) are projected before the schedule is generated therefore they are avoided during schedule generation. Resource bottleneck projection is affected by specified time windows as well as temporal loading preferences. A projected demand for each resource is calculated before any actual scheduling is done. This bottleneck information is used to affect the placement of each activity on the schedule. As each activity is placed on the timeline, the bottleneck areas are updated. This strategy allows the scheduler to move some activities away from bottleneck regions, thus allowing more activities to be scheduled that would have been unscheduled because of resource conflicts.

Closeness to loading preference: This is a relative measure of how close an activity is to it's requested position on the timeline. If an activity has requested front loading, it would be preferrable to place the activity at the beginning of the timeline as opposed to the end.

Front Loading: In the representation of the schedule, it is possible to have an activity continue into the next planning horizon. This is possible but not desirable, since less of the activity is being accomplished in the current horizon. In real life terms, one can always do today's work tomorrow, but then tomorrow's work will be delayed until the next day and so on. For this reason, it is more desirable to complete the activity within the present planning horizon.

2.2.3 Improvement Engine

A very simple iterative improver has been constructed based on the heuristic rules explained above. The word "improver" is used here to signify a better schedule, as opposed to "optimize" which implies an optimum schedule. Since the scheduler's heuristics are using incomplete knowledge of the problem, it is not "true" that the schedulers' decisions are always "optimal". In other words, even though the heuristics produce a reasonable schedule, they are by no means producing the optimum schedule. For this reason, some noise added into the decision making processes can actually produce a better overall schedule. The schedule generation process can be repeated, each time the schedule is generated, a bit of randomness is added to the decision making process. By keeping the last best schedule in memory, the scheduler can "search" for a better schedule, until the time that the schedule must be implemented.

This is somewhat similar (but opposite) to the simulated annealing approach to scheduling. In an annealer, the actual activity placement makes for a currently worse schedule but in the long run, a better schedule is produced. In the AIPS heuristics, the decision knowledge is incomplete, therefore, a worse heuristic decision may actually produce a better overall schedule when the entire scheduling process is complete.

The first time the heuristics produce a schedule, a feasible schedule is generated in a short period of time. Since the last best schedule is saved, this engine can still be considered an "anytime" scheduling engine, because at anytime in the scheduling process (after the first schedule is generated), a feasible schedule exists [Zweben 1990b]. As a counter-example, some schedulers may use so much knowledge in constructing a schedule, that they make take a significant amount of time to construct a schedule and can not be stopped in the middle of the process if a schedule is needed.

2.2.4 Non-nervous rescheduling

The same set of heuristics are also used in the process of rescheduling. The idea of non-nervous (small perturbation to the existing schedule) rescheduling is meant to save time in the outside world as well as within the spacecraft. Many other

activities, and maybe even humans on the ground may be waiting for the execution of a certain activity. If, during rescheduling, the scheduler makes large changes to the current schedule, many activities or even humans will be affected by the changes.

In some cases, the AIPS rescheduler only finds it necessary to shed a certain number of loads. This shedding is the effect of source reductions and is heavily influenced by activity priority. In other cases, certain activities may need to be deleted from the schedule, and other activities may be available to take the resources abandoned by the deleted activity.

2.2.5 Judgement of "Goodness"

Schedule judgement is needed in the iterative improver, for comparing two schedules to see which one is better, or for the user as a relative measure of "goodness". Judging a schedule is a not an exact science with many ways of looking at the problem. Of course, in a software environment, more concrete measures of schedule "goodness" are necessary. Many possibilities of judging a schedule exist, the AIPS schedule judge uses three main attributes of a completed schedule to serve as a judgment of a "good" schedule. The three attributes are power use, priority weighted number of activities scheduled, and activity position.

Power use is the amount of available power consumed by the scheduled activities. This is kept as a ratio of power used to power available. The number of loads scheduled factor is weighted by the priority of each load scheduled. It is possible to have activities continue into the next planning horizon, if this happens, the number of loads scheduled factor is penalized because the activity is not being completed in the current planning horizon. The load position factor is a measure of how close each load is to it's preferred temporal loading position.

The bottom portion of Figure 2 shows some of the schedule judgment criteria (all normalized from 0 to 1). "Time", "Counter", and "Best" have no meaning in this example. "Current" represents an overall schedule goodness rating, "Energy Used" represents the ratio of energy used in the current schedule to the energy available, "WNOL scheduled" stands for Weighted Number Of Loads scheduled, "Closeness" is a measure of how close each activity is to it's specified loading preference, and "Demand" represents the ratio of energy requested to energy available.

2.3 Interaction

There are two types of interaction; between APEX and AIPS, and between AIPS and the user. The interface between the user and AIPS is meant for testing of the scheduler, displaying information to the user, and human interaction for semi-autonomous control of the scheduler. The interface to APEX is used to implement the schedule that is generated, and to transmit rescheduling information to APEX.

2.3.1 Graphical Interface

The graphical user interface can be broken down into three major sub-views: resource, timeline, and activity displays. The resource displays are shown at the top of the screen as line graphs. The timeline is shown in the middle of the screen as a Gantt chart. The activity is shown at the bottom of the screen representing resource requests and activity information.

The scheduler has an interactive graphical user interface that can be used to test the scheduler as shown in Figure 3. This interface is fully mouse controlled and allows the user to edit activity information, resource information, as well as making changes to the schedule after the scheduling engine has given its solution. The user may test to see if they can manipulate the schedule in such a way to build a "better" schedule than the scheduling engine itself. This user interface also shows more clearly how the scheduling engine functions and makes it easier to test the scheduler. Figure 3 shows a schedule consisting of 30 activities, 2 resources, and is calculated with a time granularity of five minutes.

The upper two graphs show two sources of electrical power, with the power on the y-axis and time on the x-axis. Available power is shown as a dotted line, while scheduled power is shown as a solid line. This difference between available power and scheduled power is the unused (unscheduled) power which is shown as the dotted fill area between the available and scheduled power.

The Gantt chart in the middle of the screen shows each activity that was scheduled as a solid line or a dotted line if unscheduled. The length of the line corresponds to the length of the activity and the scale is shown on the x-axis of the Gantt chart. The earliest start and latest end points (if they are specified) are shown by brackets at each side of the activity. On the color screen, each activity is color coded by its priority. If an activity continues into the next planning horizon, this is shown as an arrow at the end of the activity.

The edit window at the bottom of the screen shows a more specific description of the activity that the mouse is currently pointing to. Values such as power demand, length, start and end constraints, priority, source, and loading preference can be specified within this window. This information is available for each activity on the schedule. In Figure 3 the mouse is pointing to the activity entitled "health", and the information for this activity is shown in the edit window.

2.3.2 APEX Interface

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 existing schedules. The initial schedule generation takes a set of resources and activities and generates a schedule for APEX to follow.

Five modification protocols exist: activity change, resource change, activity add, activity delete, and resource delete. During the execution of a schedule, the priority of an activity may change, the power demand of an activity may change, the activity may need to be dropped from the schedule, or an activity may need to be added. This change information is caused by faults in the Brassboard. Resources also may be changed during the execution of a schedule, or deleted altogether. These protocols enable APEX and AIPS to communicate system configuration information as well as reconfigure the system in the case of a fault.

APEX holds a database of activity information, as well as information on resource availability. This would probably be the job of another higher level computer in a real application, but was a suitable place for the APS project. APEX sends this activity and resource information to AIPS, and AIPS in turn generates a schedule, assigning activities start times, and subscribes to the available power.

3. FUTURE WORK

Work is currently underway in many aspects of the AIPS scheduler. The underlying representation of available resources is being changed from simple arrays to a list of start time, end time, value objects. Representation of consumable resources is being developed with a special emphasis on batteries. Work will continue on improving the heuristics used to generate the schedule and some type of reactive scheduling engine will be developed. Dynamic activity priorities and uncertainty in an activity's resource requirements will be included.

4. CONCLUSION

The AIPS scheduler has shown the ability to function in a dynamic real-world environment. Functions such as schedule generation within a time-limited environment, dynamic rescheduling, and integration with FDIR and hardware systems have been implemented in the AIPS software. Scheduler interaction with other intelligent agents has been demonstrated. These accomplishments along with the functionality of APEX and the Brassboard display automation strategies applicable to larger and more complex systems. Work will continue on APS to prove the feasibility of on-board automation with increasingly complex software systems and demonstrations.

ACKNOWLEDGEMENTS

Thanks to Todd Quinn, Tony Merolla, Walt Krawczonek, and Gene Lieberman for implementing the other portions of the APS project. Thanks to Jim Kish for his work in managing and coordinating the APS project. This work was performed under contract NAS3-25266 with Jim Kish as Coordinator.

REFERENCES

[Biefeld 1990] Biefeld, E., Cooper, L., "Operation Mission Planner: Final Report", JPL Publication 90-16, March 15, 1990.

[Britt 1990] Britt, D.L., Geoffroy, A.L., Gohring, J.R., "Managing Temporal Relations", Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence, 1990, NASA CP 3068.

[Dolce 1990]Dolce, J.L., Kish, J.A., and Mellor, P.A. "Automated Electric Power Management and Control for Space Station Freedom", In Proceedings 25th IECEC, AIChE 1990.

[Hagopian 1990] Hagopian, J., "Short Term Plan Contents", Space Station User Operations Working Group, Mission Planning Workshop, NASA Marshall Space Flight Center, May 1990.

[McClean 1990] McLean, D.R., et al., "Emphasizing Conflict Resolution Versus Conflict Avoidance During Schedule Generation", World Congress on Expert Systems, Orlando, Florida, Dec., 1991.

[Ringer 1990] Ringer, M.J., and Quinn, T.M., "Autonomous Power Expert System", In Proceedings 25th Intersociety Energy Conversion Engineering Conference IECEC, AIChE 1990.

[Ringer 1991] 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.

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

[Zweben 1990a] Zweben, M., "A Framework for Iterative Improvement Search Algorithms Suited for Constraint Satisfaction Problems", NASA Ames Artificial Intelligence Research Branch Technical Report, February, 1990.

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

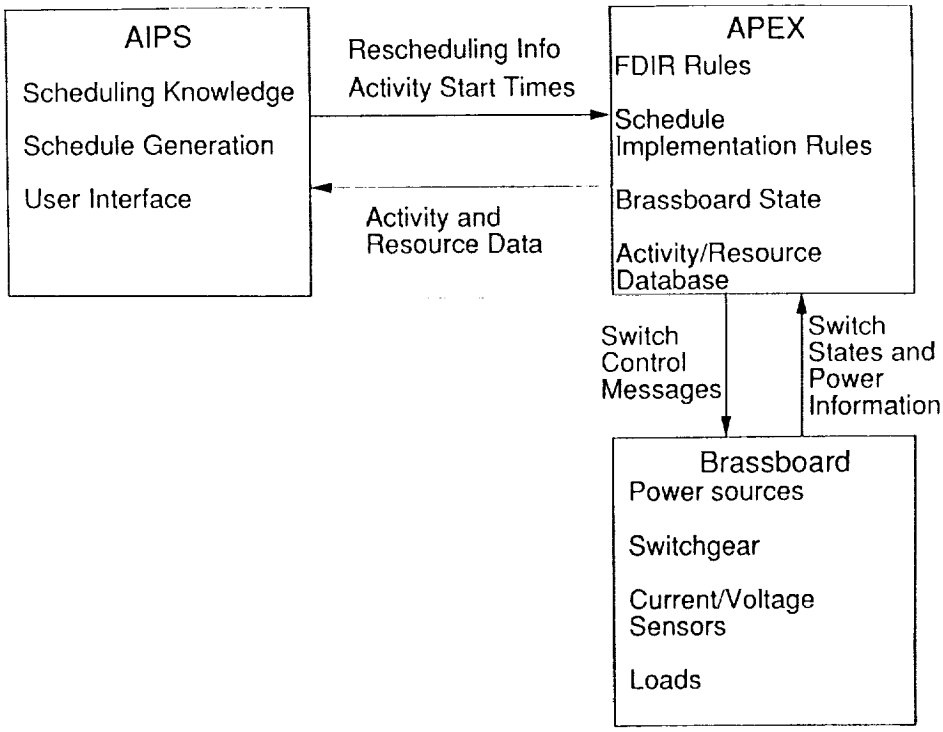

Figure 1. APS Component Functionality and Description

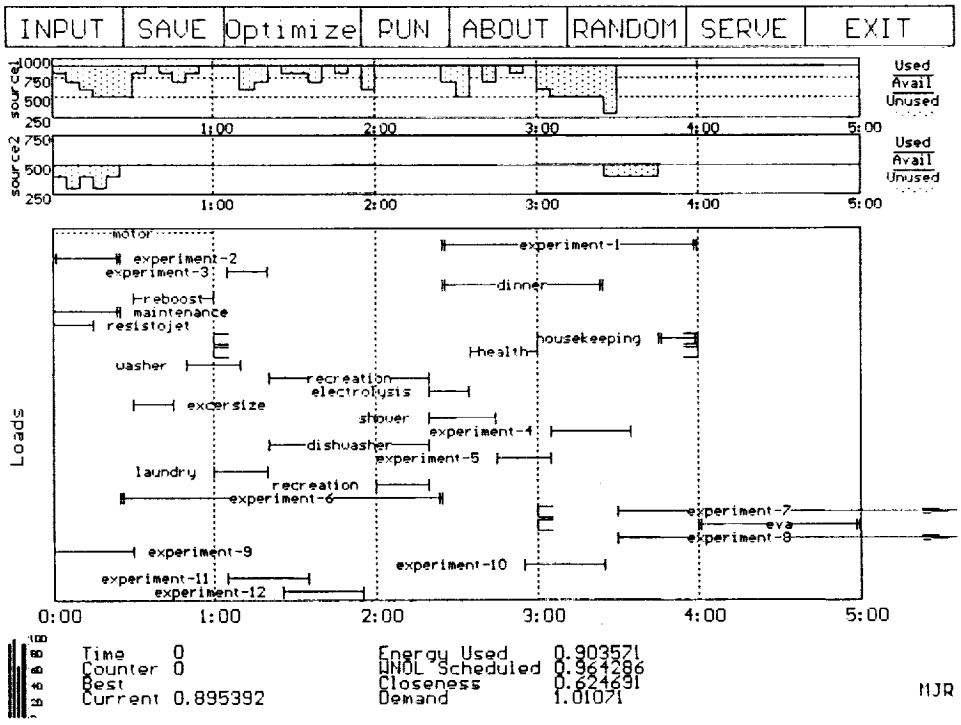

Figure 2. Generated Schedule With "Goodness" Information.

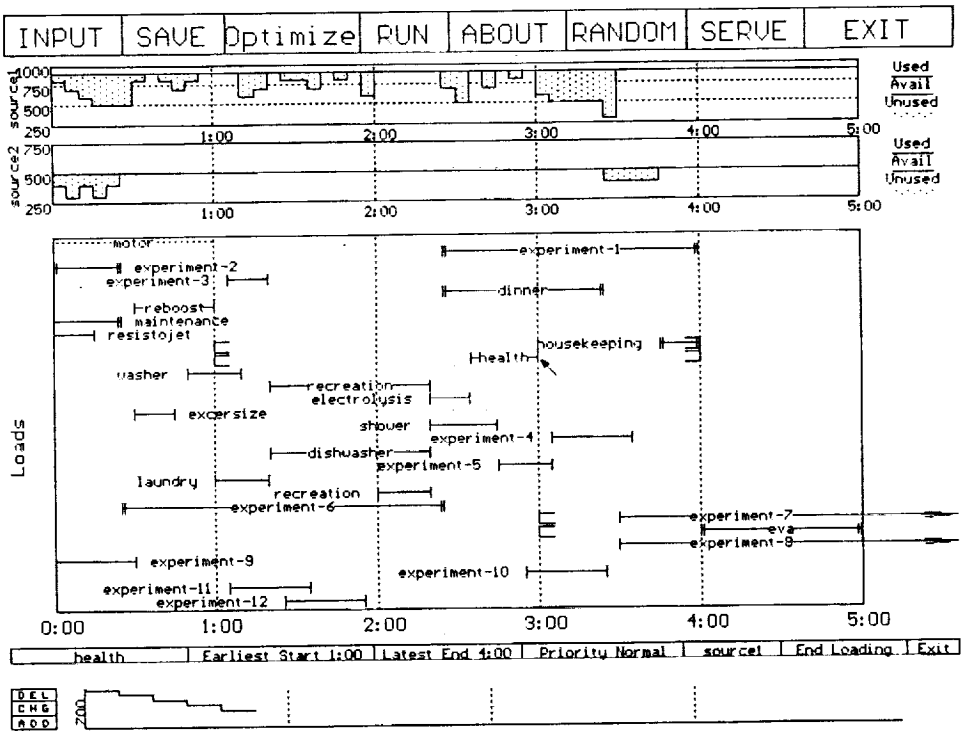

Figure 3. Generated Schedule With Activity Information.