

185

THE ROLE OF ARTIFICIAL INTELLIGENCE TECHNIQUES IN SCHEDULING SYSTEMS

Amy L. Geoffroy, Daniel L. Britt & John R. Gohring
Martin Marietta Information Systems Group
Denver, CO.

ABSTRACT

Artificial Intelligence techniques provide good solutions for many of the problems characteristic of scheduling applications. However, scheduling is a large, complex heterogeneous problem. Different applications will require different solutions. Any individual application will require the use of a variety of techniques, including both AI and conventional software methods. The operational context of the scheduling system will also play a large role in design considerations. The key is to identify those places where a specific AI technique is in fact the preferable solution, and to integrate that technique into the overall architecture.

Introduction

As Artificial Intelligence (AI) techniques have moved from the laboratory into complex applications, two things have become apparent. First, frequently more than one AI technique is required to satisfy the multitude of requirements in a large, complex operational domain. Second, AI techniques alone are either insufficient or not the most efficient means of performing these complex tasks - they must be integrated with standard software (and sometimes hardware). Scheduling in resource constrained domains is one example of this type of problem.

Resource constrained scheduling¹ is a heterogeneous problem in two ways. First, there can be tremendous differences between the characteristics of various applications' scheduling problems, even within space applications (e.g. ground processing vs. on-board experiment scheduling). Because of these differences in requirements, successful scheduling solutions for these applications will generally be somewhat different for each application. Second, there are multiple requirements within a single application. In experiment scheduling, for instance, there are requirements to limit search through the space of all possible schedules, and also to represent and manipulate quite complex resources. These are distinct subproblems, and techniques used to limit search and those used to handle complex resources will be different. Any given scheduling application may have a large number of distinct subproblems.

Scheduling within an operational context consists of both a core scheduling problem, and the problem of the scheduler's place in the operations context. The core problem

¹Scheduling problems which are not resource limited are very different in character from resource constrained scheduling and are not discussed here. The scheduling of space-based activities is resource limited, as are most ground-based operations. For the remainder of the paper the term scheduling is used to refer only to resource constrained scheduling.

will be determined by certain formal characteristics of the scheduling domain. These include considerations such as the computational complexity of the domain (as determined by the number of items to be scheduled, the structure of the items to be scheduled, the granularity of the schedule, etc.), and the predictability of the domain. The operational context drives a host of additional requirements, such as distribution of control over scheduling, human-machine interfaces, and the role of the scheduler in contingency handling.

Satisfying scheduling requirements within an operational context calls for careful examination of the specific application, and identification of the place of AI and standard techniques within an overall system architecture that address the particular characteristics of the application.

The first section of this paper introduces the resource constrained scheduling problem. The second section provides a general discussion of how AI techniques are well suited to solving many scheduling subproblems. The third section shows how the particular characteristics of specific scheduling problems will determine the appropriateness of different solution methods, with particular attention paid to AI methods. Included in this section are examples of how different applications may differ in their requirements, how a single application may consist of many subproblems, and how AI, Operations Research (OR), and other conventional software techniques may be profitably combined to provide solutions to an application's scheduling problem. The fourth

section discusses ways in which the operational context within which the scheduler will reside drives many design considerations. In the concluding section we summarize the discussion and consider how design for AI systems fit into the overall design of large, complex systems.

Scheduling in Resource Constrained Environments

Resource constrained scheduling is the fixing of activities on a timeline such that those activities may be performed at the time specified by the schedule. This entails the coordination of requisite resources, the availabilities of required ambient conditions, and the interleaving of activities which compete for resources.

Certain prerequisites must be met in order for scheduling to be feasible. Both resource availabilities and the activities' requirements must be roughly predictable. All relevant characteristics of activities and resources must be expressible. To the extent that these requirements are met, predictive scheduling - scheduling for a future time period - can be successfully performed.

For this paper, we will define three levels of description for activities: objectives, activities, and subtasks. An objective is the goal or purpose of the task, e.g. to produce a crystal. An activity is one well-specified way of fulfilling an objective, e.g. one run of the crystal growth experiment. A subtask is a step that is a part of the performance of an activity, e.g. the set-up phase in the crystal growth experiment.





TARGETING EXPERIMENT 	Resource requirements:	Power 50 Heat Rej 50	Pointing 50 Calibration 150	Data Collect 190	Shut Down 50	
	Conditions:	Target must be available for data collection phase Vibration must be < x for calibration and data collection Atmospheric pollution must be < y Side effects: Creates vibration when rotating in pointing phase Time windows: Mon. - Fri. 11 am - 6 pm (when ground support is available)				
SPIDER EXPERIMENT 	Resource requirements:	Continuous power & heat rejection, 25w each 1 Crewmember (PO or PS) for observation phases				
	Conditions:	Observation phases vibration < z Observations must occur at least 10 minutes after "sunrise", during "daylight" only Coordination: Observation phase of this experiment should co-occur with the filming phase of the Public Relations filming				
CRYSTAL SAMPLE GROWTH EXPERIMENT 	Resource requirements:	Crew 1 Power 0 Heat Rej 0	Set-up 1 Heat 500 200	Grow 0 250 500	Centrifuge 1 110 110	Analysis 1 25 25
	Conditions:	Sample growth < p vibration Sample analysis < q vibration Gasses and liquids: 20l Helium 5l Argon 2l H ₂ O Side effects: Centrifuging induces X vibration				
PUBLIC RELATIONS FILMING 	Resource requirements:	Crew 1 Video Cam 1 Power 0	Set-up 1 Film 1 125 1 roll film	Break-down 1 1 0		
	Conditions:	No venting during filming Filming < x vibration Coordination: Filming phase must co-occur with spider obs				

Figure 1. A simplified example of activities which might require scheduling for Space Lab, described in terms of their requirements.

An example will be used to illustrate some characteristic problems commonly found in scheduling applications. This is not intended to represent a "generic" scheduling problem. As we shall argue later, applications vary enormously in the ways their scheduling problems may be characterized, and these differences preclude the possibility of a single representative scheduling problem. Rather, the example serves to illustrate one typical version of a scheduling problem.

Fig.1 shows a simplified example of a hypothetical scheduling problem. Suppose that three experiments - one using a targeting instrument, another involving a crew member's observation of the activity of some spiders, and another involving the generation and centrifuging of some samples - and a public relations filming activity, all require scheduling for Space Lab .

For each of these activities certain resource requirements must be met: power (for the targeting and sample experiments), crew time (for the spider and sample experiments), etc. Insofar as these resources are limited, different activities may compete for these resources, requiring coordination of the activities. Conditions for each of the experiments also must obtain - the targeting instrument must be able to acquire its target, and may require a minimum vibration, while the centrifuge phase of the sample experiment may generate a certain amount of vibration. Additionally, there may be a requirement to film a crew member performing the observations in the spider experiment for a publicity film, and this will require

coordination of the filming's timing, resource and conditions requirements with those of the spider experiment. Because time and resources in space are rare and costly it is of the utmost importance to generate as efficient a schedule as possible. The process of producing an efficient schedule which provides the necessary coordination of resources and conditions for these activities can be quite complex.

There are several sources of difficulty in this scheduling problem which are found in many different scheduling applications. The scheduling objects, i.e. the activities and resources, can be quite complicated. The relations between these objects can also be complicated. Ensuring that these objects and their relationships are all appropriately represented, and that there are means of reasoning about them which will allow even the verification of schedule validity is no trivial task. However, the most critical challenge is controlling the combinatorics of the problem - producing a good schedule given all the possible schedules that might be generated.

Combinatorics. It has been demonstrated that procedures which guarantee an optimum solution to scheduling problems are either NP-complete or NP-hard (Ibaraki & Katoh, 1988), depending on the characteristics of the particular problem. Realistic scheduling problems are most frequently NP-hard. What this means is that the computation time for finding an optimal solution to a scheduling problem increases exponentially or worse as a function of the number of subtasks and resource disjunctions (choices between unique resources in a resource pool which satisfy the

resource requirement) under consideration.

In the example given in Fig. 1 above, each of the phases of each experiment counts as one subtask (e.g. the targeting experiment consists of four subtasks - pointing, calibration, data collection, and shut down) yielding 14 subtasks. One three-valued resource disjunction is represented in the choice between crewmember types. For any given subtask requiring crew time, you may use any one of three types of crewmembers - Payload Operators (PO's), Payload Specialists (PS's), or Mission Specialists (MS's), unless some restriction is specified. Multiplying out each subtask by the resource disjunctions found in each subtask, we find that $n=27$. This, however, represents only a small subset of a realistic scheduling problem for the Space Lab application. In a real application, there are many more experiments under consideration, each experiment consists of more subtasks, there are more resources and there are more disjunctions of those resources .

Parunak (1987) states, "... consider a problem that in the worst case requires 2^n microseconds to solve, where n is the size of the problem [a] problem of size 10 will require no more than .001 seconds to solve... of size 40... as long as 12.7 days ... of size 60, 366 centuries! ... Even with a thousand fold increase in speed, the size of problem that we could guarantee to complete in 366 centuries increases only to 70."

These increases in computation time are a direct reflection of the increasing size of the space of possible schedules - all possible combinations

of placements of subtasks on the schedule. Most realistic, complex scheduling problems are not amenable to optimization techniques simply because of the combinatorics involved. Thus, methods must be derived to arrive at solutions without searching the entire space of all possible schedules.²

The Suitability of AI Techniques for Scheduling Problems

There are a number of characteristics of scheduling problems that make them amenable to AI solutions, some having to do with the combinatorics problem, others having to do with other aspects of scheduling - requirements for rich representation techniques, exploitation of constraints, planning problems, and the utility of expert knowledge.

Combinatorics. One of the main goals in the development of AI as a field has been to devise methods which

² An alternative approach to this problem has been developed in Operations Research (OR). There has been a great deal of work in OR on algorithmic methods which yield good, but not optimal solutions to scheduling problems, but these are also limited, either by size of the problem that can be reasonably attacked by these methods, or by the limiting assumptions that must be made for the methods to be effective (Graham, 1978). Also, these techniques require an "objective function" - a formula which supports precise measurement of schedule value which the algorithm tries to minimize or maximize. In many real scheduling applications, there is no objective function. Human schedulers cannot formulate a precise mathematical combination of their many often conflicting goals (e.g. resource efficiency, relative priorities for activities, fairness) that represents the value of a schedule. However, these methods may prove quite valuable if used on small subproblems for local decision making.

circumvent the problems of combinatorics by intelligently guiding search through enormous spaces, and these methods are applicable to scheduling problems. (In fact, a number of these methods have been adopted by OR practitioners and others, and are now considered standard computer science techniques).

Representations. Many of the objects in these domains may be considered semantically rich, hierarchically or heterarchically structured, with many different types of characteristics (consider the description of the Space Lab experiments given above). The representation techniques developed for AI, particularly object-oriented programming, are ideally suited for this domain.

Constraints. There are many constraints that must be respected, or which can be exploited in developing a schedule, and constraint propagation and relaxation techniques have been well developed in the field.

Planning. Some aspects of the scheduling problem turn out to be problems in planning - in some cases an objective may be specified, but the actual activity to support that objective may be underspecified. The unique specification of the activities which will achieve the objective is a planning problem, another area in the AI field.

Expertise. Much of the know-how in scheduling is the purview of human experts, and may be amenable to expert system techniques.

There are other characteristics of scheduling which make it a good candidate for AI techniques, but these examples suffice to show that AI is a

good path to explore in creating solutions.

Matching Techniques to Scheduling Problem Characteristics

The discussion above provides a general picture of how AI might be applied to scheduling problems, but little specific about how to actually apply these techniques. This is because scheduling is a heterogeneous problem. The exact form of a solution to any given scheduling problem depends on the characteristics of the particular problem. A sample of relevant characteristics are:

- I) Activities
 - A) Total number
 - B) Complexity
 - i) number of subtasks
 - ii) number of resources/subtask
 - iii) dependencies between resource requirements
 - C) Similarity of activities
 - D) Fixedness
 - i) Timing
 - ii) Number of different ways of achieving the same objective
 - E) Fragmentability
 - F) Co-dependencies between activities or subtasks
- II) Resources
 - A) Number
 - B) Disjunctions
 - C) Complexity
 - D) Similarity of resources
 - E) Co-dependencies between resources
- III) Time
 - A) Granularity
 - B) Schedule duration relative to activities' and subtasks' durations

- IV) Schedule
 - A) Repetitiveness
 - B) Resource costs
 - C) Activities' values
 - D) Goals
 - i) explicitness
 - ii) number
 - iii) types
 - iv) variety
- V) Methods used in current operations
 - A) Adequacies/Inadequacies
 - B) Experts
 - i) existence
 - ii) quality of performance

This list is not intended to be an exhaustive catalog of all relevant factors, but it represents many important ones. For instance, the number of subtasks (IBi) and the number of resource disjunctions (IIB) will determine the size of the combinatorics problem. The complexity of activities (IB) and resources (IIC) will determine the complexity of the representations used. Number of ways of achieving the same goal (IDii) indicates the extent to which a planning problem exists.

The values of these characteristics can vary considerably between different scheduling applications. The specific characteristics of a given scheduling application will determine what combination of techniques, both conventional software and AI methods, are most appropriate. There are three major points to be made here. First, different applications will demand different solutions. Second, because real scheduling applications are really a combination of a number of thorny problems, a single application will probably require more than one methodology. Third, these methodologies may often be a

combination of AI and conventional software methods.

Differences between diverse applications' characteristics will demand different solutions. A few examples serve to show how the different characteristics of a problem can help to determine appropriate solution methods. For instance, in domains where there are many co-dependencies between activities, between subtasks or between resources, constraint propagation techniques will be extremely important. Search methods that evaluate current state as a function of schedule "goodness" so far and/or projected distance to goal (e.g. best-first search, genetic algorithms) work well in domains where a) scheduling goals are explicit, b) their values may be defined quantitatively, c) a function defining the combination of these values may be defined to reflect an overall schedule value, and d) these values are easy to measure on an existing schedule, but not otherwise. Where human experts perform the scheduling function quite well, an expert systems approach would generally work well, but not in domains where human scheduling is considered inadequate.

An interesting comparison of matching techniques and domain characteristics is shown in contrasting scheduling for some Deep Space Network (DSN) problems and scheduling the Laboratory experiments onboard Space Station. The DSN problem can be characterized as having an extremely large number of activities, each fairly simple - consisting of one or few component subtasks, and requiring a small number of resources. Also, the activities to be scheduled are very similar to one another, using basically

the same resources in basically the same ways. In contrast, there are fewer activities to be scheduled for the Space Station Laboratory, but each of the activities is more complex. Each activity consists of a larger number of subtasks, each requires a large number of resources, and the activities are much less similar to each other.

A scheduling strategy adopted for the DSN problem might be to create an initial schedule paying little attention to resource limitations, and in which resources are overbooked, then to shuffle activities on the schedule to try and ameliorate the overbooking. The primary strategies in use here are backtracking and evaluation of entire candidate schedules. In contrast, for the Space Station Laboratory problem, a scheduling system might use constraint propagation techniques and a number of intelligent heuristics to create an initial schedule which is conflict-free. These methods would concentrate on finding places on the schedule where each activity fits without resource or other constraint violations. The methods used here would be constraint propagation and local optimization. These two different approaches each work well for their intended applications.

Consider trading the approaches between the applications. To fully appreciate the implications of this trade, it helps to consider the characteristics of the different domains more abstractly. Imagine activities to be n -dimensional shapes that must be packed as tightly as possible. One dimension represents time, and each of the other dimensions represents a resource required for the activity. The space into which the activities must be

packed will have the same number of dimensions as the total number of unique resources used by all the activities, plus one dimension for time. Figure 2 shows a representative simple case of a four-dimensional task.

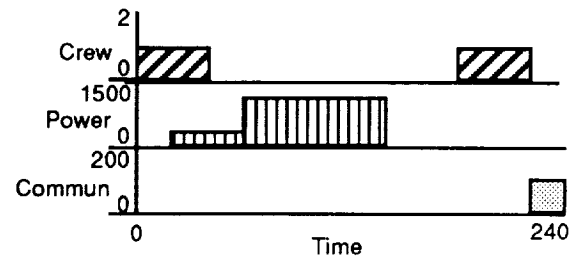


Figure 2. A task requiring three resources, crew time, power, and communications time, against time. This translates into a four-dimensional shape for the task.

Because the DSN activities are fairly simple and similar, the shapes for each of the tasks will be much alike. Because there are few resources under consideration, the n -dimensional space into which they will be packed has a small number of dimensions. In contrast, the Laboratory tasks are much more complex and less similar, so the task shapes will be very different from each other. The n -dimensional space into which they must be fit has a large n (over 200 resources are used in scheduling for one version of this problem).

The approach of initial random placement followed by shuffling to ameliorate overbooking is good for cases where the shapes are basically similar, because the shapes representing each task are pretty much interchangeable. Random placement and shuffling are dreadful strategies, however, when the shapes of activities are very different, and there are a large number of

dimensions to be matched. This is because (except in a resource-rich environment) random placement is unlikely to yield viable placements for complex activities. Further, randomly substituting an activity which is currently an unsuccessful fit for one that is a successful fit only works insofar as the successful item is blocking the resources needed for the other item. The more dimensions under consideration, and the more different the two activities are on those dimensions, the greater the probability of failure.

The approach which focuses on goodness of fit for activities at each step in schedule generation is good for scheduling dissimilarly shaped activities in a large number of dimensions because it focuses on finding such fits. However, it is computational overkill to use this approach for shapes which are highly similar and are to be fit in a small number of dimensions. Most of the decision making strategies used in creating an initial schedule would be irrelevant in the DSN application. Also, since this approach does not support random modifications (a good idea for DSN-type tasks), it fails to take advantage of some simple strategies that, for the DSN domain are effective at reaching better quality schedules quickly.

Different methods will be used for any single application. Any given scheduling application may consist of a combination of hard problems. Some of these problems may interact, so that solutions must be co-designed, while others may be independent, and the design for solving the different problems may be performed independently.

The example used in Fig. 1 illustrates a few of these types of problems. There are several different kinds of constraints that must be respected in order for these experiments to be successfully scheduled. Two of the major constraints are resource requirements (e.g. power and crew requirements) and temporal co-dependencies (e.g. do the public relations filming while an astronaut is observing the spider experiment). These two constraint types require different aspects in representation, and different computational methods to ensure that these constraints are met. However, both constraints are used to compute answers for a single problem - where an activity may be placed that meets all of its constraints. Because of this, their design must be tightly linked.

In contrast, there are other difficult aspects of this scheduling problem that are fairly independent of the computations for temporal and resource constraints. The management of several of the resources in this scenario is complex. Crew, for instance, has certain restrictions not only on total amount of time worked in a shift, but on the combination of experiment types that may be successively scheduled, preferences for different activities, relative experience with and qualifications for different activities, etc. Data transmission is a complex combination of real-time transmission and storage with subsequent transmission, where transmission times may be determined by what items appear on the final schedule. These resources require complex management systems themselves, which may take on a variety of forms (e.g. expert systems, OR methods, etc.) depending on the details of the

management problem. The design of each resource manager, however, is nearly independent of the constraint propagation methods for temporal and resource constraints - the only requirement is that the manager provide the type of information about the resource required for scheduling.

The best solutions may combine AI and more conventional software techniques. Because any scheduling application consists of a variety of problems, it will generally be the case that some of the solutions will consist of conventional software methods, and some will use AI techniques.

Many of the examples given above have illustrated the utility of AI solutions to problems in various applications. The most obvious example of an area for conventional software are parts of the scheduling process which are straightforward and algorithmic. Most applications, for instance, require a good deal of bookkeeping - tracking what resources and conditions are available where on the scheduling timeline, and updating those availabilities as the scheduling process proceeds. This is best suited to conventional computing procedures.

There are also interesting possibilities for combining conventional and AI techniques to attack the same problem. Several recent systems (e.g. Berner, Durham & Reilly, 1989) have combined AI and OR techniques. It is possible, for instance, to use a heuristic method to decompose the scheduling problem into subproblems which are more tractable, and then use OR or optimization techniques to solve the subproblem (Britt, Geoffroy & Gohring, 1990). It also might be promising to

extend the multi-perspective scheduling strategies used in OPIS (Smith, Fox and Ow, 1986) to a multi-technique strategy, where based on the current state of the problem (e.g. whether resource bottlenecks appear, how large the current search space is, etc.) different scheduling techniques might be selected, and these techniques might be a variety of AI and OR techniques.

The Operational Context

The scheduling problems described above are realistic, and solutions to those problems can be embedded into real operations. The scheduling problem as discussed so far may be considered the core scheduling problem. Planning and scheduling in the operational context, however, entails much more than just the core scheduling problem, and the larger operational context complicates the requirements for a scheduling system. Some of these additional requirements may be addressed by systems which are entirely separate from the scheduling system. So for instance, some kind of support to help users formulate activity definitions will be required. A separate activity editor can be implemented to fulfill this requirement. Such an editor would be a good candidate for intelligent human-machine interface techniques. The design of the core scheduler will remain untouched by the requirements for the activity editor. However, there are other requirements levied by the operational context that must be reflected in the design of the core scheduler itself.

Three major complicating factors will be discussed here. First, the scheduler performs a function which must be integrated with the other

functions of the larger planning and scheduling system. Second, in the operational world, nothing ever really goes exactly as planned - there are always contingencies popping up which invalidate a schedule and require some reaction to get the schedule back on track. Third, the planning and scheduling process changes in character over time, from long-range planning to the scheduling of today's activities.

Scheduler integration. In many operational contexts, a scheduling system will reside as one node in a complex network of hardware and software systems and human users.³ The scheduler will be involved in numerous information exchanges, e.g. receiving data about resource availabilities, or transmitting data about activities' timings. Ensuring that the design of the scheduler can support such exchanges may be time consuming, but will have little impact on the technical approach to the core scheduling problem.

However, other aspects of interactions with the other nodes in this network may drive some design decisions related to the core scheduling system. Two main issues here are control of the scheduling process, and the degree to which the actions of the scheduling system can be understood.

It is unlikely that a scheduling system for a complex scenario will always be run entirely autonomously. Even if it is feasible, full autonomy is probably

not desirable. Human operations managers will want to have the final authority on decisions regarding the schedule, even if this authority is only rarely exercised. Operators must be able to control and interact with the scheduler. This places demands not only on the design of the user interface, but on the control structures of the scheduling system, particularly if the goal is a user-determined level of autonomy. It also requires that the system operate in such a way that people can understand what the system is doing.⁴

Control of the scheduler may not be limited to interactions with a single user or single user type. The scheduling of unmanned platforms such as the Earth Observation System (EOS) for instance, involves a network of science users, platform managers, instrument managers, and communications (TDRSS) managers, and all are involved in the scheduling process. Each has a different realm of authority, and each can control the scheduler in different ways. The scientists interact with the scheduler in terms of their experiments, while the scheduling concerns of the platform manager are to provide platform resources, such as power, to support the schedule, and to schedule activities which maintain the health and safety of the platform. These users interact with the scheduler asynchronously, which has implications for both the control structure and the scheduling strategies of the system. Because users interact with the scheduler

³ There are a number of functions (other nodes) in this larger planning and scheduling context which are good candidates for AI applications, but space limitations preclude pursuing this further here.

⁴ This does not necessarily require that the system reason like an expert - people can understand, for instance, that a thermodynamic model is used, even though they don't create schedules manually using that method.

asynchronously across long periods of time, the information on which scheduling is based is always in a state of flux. The scheduling strategies of the system in this scenario have to reflect the tentative state of the scheduling information available at any given point in time.

Contingencies. Schedules are based on assumptions made about activities' requirements, the resources that will be available for the scheduling period, conditions which will be true, etc. After a schedule has been generated, some of these assumptions will turn out to be false. There may be a power failure, some equipment might break, an activity might take longer than projected, or some new, previously unscheduled activity may need to be forced on the schedule. In most of these cases, the schedule will be invalidated.

The design of the scheduling system should support the ability to reformulate the schedule based on the new information. In general, it is not desirable to generate an entirely new schedule, but to repair the existing schedule, so techniques which modify existing schedules are required. Because many of these contingencies will happen during scheduling execution, reactive rescheduling must be fast. If the same techniques are used for scheduling and rescheduling, then the scheduling strategies must be designed with the speed issue in mind. If different systems are used, then careful attention must be paid to ensure that the output of the initial scheduling process can support the input requirements for the rescheduling system.

The planning and scheduling process across time. Mission operations

planning and scheduling for applications such as the Shuttle or Space Station Freedom begin years ahead of the actual mission. The different phases of this process, from the strategic planning, years ahead of time; to the short term scheduling, just shortly before execution; to near real time reactive rescheduling, have different characteristics, and different requirements. The length and granularity of the timeline are different for each of these phases, and the level of detail about the activities to be placed on a timeline are also different.

This is actually an expansion of the point made earlier, that any given scheduling application has a multitude of requirements. In this case, entirely different systems may be needed to accommodate each phase of the planning and scheduling process. In addition to creating systems for each of these phases, ways of transitioning information between the phases is required. This may require some consideration of commonalities for data structures between the system, and may require that the strategies used in each phase are complementary or synchronized.

Conclusions

What can AI do for scheduling applications? As pieces of an overall system architecture, AI techniques can be used quite successfully. Methods for handling various types of computational complexity have been well explored. Techniques for representing complex objects and relationships, constraint management, planning, and representation of expert knowledge and methods are all areas of strength in the field of AI. Intelligent human-machine interface techniques can be used profitably to help systems fit into operational contexts. The key

is to identify those places where a specific AI technique is in fact the preferable solution, and to integrate that technique into the overall architecture.

Much of the discussion in this paper addresses issues that are not unique to the development of AI systems, but which are pertinent to any software development for a complex functional system. The points about matching techniques to problem characteristics and taking into consideration the operational context of the scheduling application have to do with good system design in general, rather than design issues unique to AI. This is as it should be. As AI enters mainstream use, AI techniques become another set of methods in the repository of software solutions. The flip side of this is that AI systems are not immune to the problems associated with design for complex applications. Detailed problem analysis is the only way to find a good match between these techniques and the applications. The specifics of the problem, and the operational context into which the system is embedded must help to determine the form of the solution proposed.

References

Berner, C.A. Durham, R. & Reilly, N.B. (1989) Ground data systems resource allocation process. *Goddard Conference on Space Applications of Artificial Intelligence*. Greenbelt, MD. NASA Conference Publication 3033, 37-47.

Britt, D.L. Geoffroy, A.L. & Gohring, J.R. (1990) Managing temporal relations. *Goddard Conference on*

Space Applications of Artificial Intelligence (these proceedings).

Graham, R.L. (March, 1978) The combinatorial mathematics of scheduling. *Scientific American*. 124-132.

Ibaraki T. & Katoh, N. (1988) *Resource allocation problems: algorithmic approaches*. Cambridge, MA: MIT Press.

Parunak, H. (1987) Why scheduling is hard (and how to do it anyway). *Proceedings of the 1987 Material Handling Focus*. Georgia Institute of Technology.

Smith, S.F. Fox, M.S. & Ow, P.S. (Fall, 1986) Constructing and maintaining detailed production plans: investigations into the development of knowledge-based factory scheduling systems. *AI Magazine*, 45-61.

