DISTRIBUTED PROJECT SCHEDULING AT NASA: REQUIREMENTS FOR MANUAL PROTOCOLS AND COMPUTER-BASED SUPPORT

Final Report

NASA/ASEE Summer Faculty Fellowship Program--1992

Johnson Space Center

Prepared By:	Stephen F. Richards, Ph.D.
Academic Rank:	Associate Professor
University & Department:	Ambassador College Computer Information Systems Department Big Sandy, Texas 75755
NASA/JSC	
Directorate:	Information Systems
Division:	Software Technology
JSC Colleague:	Chris Culbert
Date Submitted:	August 7, 1992
Contract Number:	NGT-44-005-80

ABSTRACT

The increasing complexity of space operations and the inclusion of interorganizational and international groups in the planning and control of space missions lead to requirements for greater communication, coordination, and cooperation among mission schedulers. These schedulers must jointly allocate scarce shared resources among the various operational and mission oriented activities while adhering to all constraints. This scheduling environment is complicated by such factors as the presence of varying perspectives and conflicting objectives among the schedulers, the need for different schedulers to work in parallel, and limited communication among schedulers. Smooth interaction among schedulers requires the use of protocols that govern such issues as resource sharing, authority to update the schedule, and communication of updates. This paper addresses the development and characteristics of such protocols and their use in a distributed scheduling environment that incorporates computer-aided scheduling tools. An example problem is drawn from the domain of space shuttle mission planning.

INTRODUCTION

Scheduling is the process of assigning resources and times to each activity of a plan (or plans) while ensuring that each constraint is obeyed. Optimization criteria can determine the relative desirability of two alternate schedules. Although scheduling problems are often simple to visualize and express, scheduling is an NP-complete problem, so attempts to apply mathematical programming to scheduling have met with very limited success [2, 6]. In fact, programming approaches have been limited to very narrow problem domains, especially that of the job-shop, in which jobs must be assigned to various machines.

This paper focuses on the class of scheduling problem in which:

1. activities have precedence relationships (one activity must not begin until another activity has completed);
2. resources are limited;
3. objectives or optimization criteria exist that may be used to rank competing schedules; and
4. the time frame in which to complete all activities (or as many activities as possible) is limited.

This class of problem differs from the job-shop problem domain in that a job-shop problem assumes an infinite time line in which all activities may complete. In a job-shop problem, all activities are scheduled regardless of the total time required. In contrast, in this paper resources may be over-subscribed, so that even the optimum schedule might not accommodate all desired activities within the time limitations. Thus, provision must be made for selecting between competing activities (or sets of related activities) where insufficient time exists for the completion of all activities.

To assist users in developing viable schedules, NASA has developed COMPASS (COMPuter Aided Scheduling System) [3], a computer-based tool that interactively schedules activities in a user-specified order. COMPASS provides graphical tools for displaying activities, resource availability, and schedules. An activity defined in COMPASS may have precedence requirements and require resources. Activity attributes supported by COMPASS include priority, required resources, duration, earliest permissible start time, latest permissible end time, and state conditions. COMPASS has enjoyed widespread acceptance and use within NASA and the contractor community.

NASA has recently proposed enhancing COMPASS to support multi-user or distributed scheduling problems. This paper focuses on the issues raised by distributed scheduling and on requirements for computerized support of this problem domain. The next section of this paper defines distributed scheduling and addresses these issues. This is followed by a discussion of

some of the human issues involved in the development of protocols for use by multiple teams of schedulers who must cooperate to produce joint schedules.

DISTRIBUTED SCHEDULING

Definition

Distributed scheduling consists of those scheduling problems involving:

1. several schedulers,
2. who can work independently,
3. each of whom is responsible for scheduling separate sets of activities that are somehow interrelated, and
4. must share a common pool of resources.

Besides sharing a common resource pool, the activities may also have precedence requirements, or one activity may establish a state that another activity requires, etc. While the schedulers may work independently, the need to coordinate the interactions among their tasks prohibits purely independent work. Distributed scheduling problem domains of particular interest to NASA include the scheduling of astronomical satellite experiments, personnel training, and space mission activities.

The interactions among schedulers can be cooperative or competitive. Cooperative scheduling is defined as those cases in which:

1. All schedulers have the same objectives;
2. Responsibility for scheduling has been divided in order to share the labor; and
3. Protocols serve primarily to coordinate and synchronize.

In large problems the size and complexity of the scheduling task and the limited abilities, skills, knowledge, and resources of any individual make the distribution of the scheduling task a natural and necessary means of developing the required schedule. By distributing the work, each scheduler can concentrate on a manageable volume of work in a narrow domain. Some schedulers may develop specialized knowledge and skills suitable only to their particular domains.

In contrast, competitive scheduling consists of those cases in which:

1. Each scheduler has his or her own objectives;
2. The necessity of sharing common resources interferes with the simultaneous achievement of these objectives;
3. The pursuit of individual objectives leads to competition for the common resources; and
4. Protocols serve largely to arbitrate competition by allowing all schedulers fair access to

shared resources.

Competitive scheduling can arise in situations in which there are contractual agreements among different parties or in which different resources are owned by different groups. Such situations can dictate the division of responsibility for scheduling among several groups, with each group having its own set of goals.

General Discussion of Goals

The lack of a single point of control increases the complexity of the overall scheduling problem (as a result of the necessary communication overhead) and raises several issues regarding the interactions of multiple schedulers and the integration of their individual schedules. The most basic issue raised by distributed scheduling is that of goals. What measure of goodness is most appropriate in a distributed environment? How do the optimization criteria for a distributed scheduling problem differ from those for a non-distributed problem? Variables commonly used for scheduling problems include [4, 5]:

- Completion time: the time at which processing of the last activity completes.
- Flow-time: the total time that activities spend in the shop.
- Lateness: the difference between the completion time of an activity and some pre-specified due date associated with that activity.
- Tardiness: equal to lateness when lateness is positive, otherwise equal to zero.

Schedule evaluation criteria typically involve minimizing or maximizing the mean, total, minimum, or maximum of one or more of these variables. In a standard job-shop problem, these criteria are assumed to be universally agreed upon. However, even in such a standard, non-distributed scheduling environment, the various tasks to be scheduled may belong to several different customers (perhaps represented by members of the marketing staff), each of whom would prefer that his or her tasks be given high priority. Thus, even in a non-distributed setting conflicting goals may exist. When conflict exists, the scheduler must have some means of determining a set of priorities to be applied to the scheduling task. The scheduler may be flexible in his or her choice of priorities, adjusting them to the needs of the moment. For example, the scheduler might attempt to mollify a major customer who has previously been slighted by giving preference to that customer's work in the current schedule. Regardless of the conflicting demands, however, the optimization requirements are formulated under a single point of control and this procedure can succeed because the single scheduler (or team of schedulers) who develops the optimization criteria also controls the entire resource pool.

In a distributed schedule, however, individual schedulers must share resources, so one scheduler optimizing his or her schedule may restrict another scheduler's options, resulting in a subopti-

mal global schedule. The issue of a global measure of goodness becomes more important in distributed scheduling than in individual scheduling. This is true because an individual scheduler can accept a schedule even without a specific measure of goodness; the schedule may balance several conflicting needs fairly and "just look good." A distributed schedule, in contrast, must "look good" through several sets of eyes. When a team of schedulers must continue to work together on future projects, perceptions of inequity or misplaced priorities can engender resentments that will poison these on-going relationships. Thus, some mechanism for balancing both local and global optimization must be provided. The protocol used by the schedulers to coordinate their activities must support optimization techniques that are perceived as both equitable and efficient.

Requirements for Competitive Scheduling

NASA needs to develop protocols that facilitate the development of successful schedules in "competitive" distributed environments that generally satisfy the objectives of the separate schedulers. This requires protocols that govern the process of building the schedule as well as protocols that govern how conflicting objectives are resolved. Selecting a desirable scheduling protocol requires balancing several possibly conflicting requirements, including the following [1]:

1. The protocol should encourage the development of high quality schedules that score well when evaluated by either the global optimization criteria or the optimization criteria of individual schedulers. Where conflicting objectives exist, the protocol should lead to a reasonable compromise.
2. The protocol should be easy to understand, use, and implement. Features enhancing ease of use include ease of learning; minimum complexity; informative to the user of the state of activities, resources, etc.; and natural representation of concepts. Yet the process should be sufficiently rich in features and notation to encompass a wide range of scheduling problems.
3. The protocol should be mechanical and unambiguous.
4. The protocol should be general enough to work with a wide range of scheduling software. Schedulers should not be constrained to use a particular scheduling system or even the same system.
5. The resulting schedules should be resilient to unexpected changes.
6. The overhead should be kept to a minimum. For example, the volume and frequency of communications should be low.
7. The time required to develop schedules should be short, especially in highly dynamic environments.

8. Any computerized support should have a short response time. This requires that optimization techniques be computationally simple.
9. Rescheduling (the repair of a schedule because of unexpected occurrences, such as delays and loss of resources) must be especially fast.

Several sample scheduling techniques are listed below. The alternatives are discussed in terms of division of resources, communication, cooperation, and optimality.

1. Schedule tasks by priority. This approach requires that all tasks be known and prioritized in advance and then be scheduled in priority sequence. This is really non-distributed scheduling, except that we have several schedulers responsible for collecting tasks and we may provide improved computer support to enable the individual schedulers to track their own set of tasks by viewing only their portion of the schedule. This protocol also requires some mechanism for assigning priorities to tasks, such as a central authority or a voting scheme. (Schedulers with conflicting objectives may never agree on the assignment of priorities.) Although participants may perceive this method as fair (since no lower priority activity will be scheduled while a higher priority activity remains unscheduled), following this method strictly does not allow for compromises, such as scheduling two medium priority, low resource intensive activities instead of one higher priority, high resource intensive activity.
2. First come, first served. In this approach all schedulers are equal and none has priority over the others. Resources are not assigned to individual schedulers, but may be reserved by any scheduler. No cooperation among schedulers is required. Optimization is poor, because no attempt is made to balance the needs of multiple schedulers. There is a tendency among schedulers to reserve resources early, even before they know their full requirements. This hoarding can result in the allocation of resources to low priority tasks.
3. Divide resources among schedulers in advance. This method permanently allocates resources to specific schedulers who can use them as they choose. No communication or cooperation among schedulers is required. Schedulers need not even know the global schedule. This approach is impractical when there is a potential state conflict between tasks (e.g., when two schedulers independently schedule a treadmill experiment and a microgravity experiment that requires no vibration). This approach may also yield poor schedules when one scheduler assigns resources to low priority tasks or leaves resources unused that could be used by another scheduler. In this approach the quality of the resulting schedule is limited by the appropriateness of the initial allocation of resources. A poor allocation may result in few activities being successfully scheduled.

4. Divide resources among schedulers in advance but permit borrowing. This approach differs from the previous one by permitting schedulers to negotiate among themselves to improve their schedules. There is still no need for global optimization criteria. The status and bargaining power of individual schedulers is determined by the initial allocation of resources. Communication needs consist of a knowledge of resources available to other schedulers.
5. Sharing of intentions among schedulers. In this approach schedulers review their intentions with their peers and receive feedback before reserving resources and committing to a particular schedule. While this approach has the potential for producing high quality schedules through the sharing of knowledge and expertise, it also imposes a heavy communication burden among schedulers that can negate much of the benefit resulting from distributing the scheduling task. This approach is also fragile in that its success depends on the voluntary cooperation of each scheduler. Where this cooperation fails, this approach can degenerate into a first come, first served system.
6. Simultaneous iterative scheduling. In this method each scheduler devises a schedule and shares it with others. Schedulers identify and resolve conflicts by some agreed upon method. If unscheduled tasks and unallocated resources remain, another round of scheduling follows. In this approach all schedulers must be ready to schedule simultaneously. Also, each participant must be provided some incentive to cooperate with the others in resolving conflicts. The global schedule must be available to all schedulers.
7. Consecutive iterative scheduling. In this method the schedulers are divided into two or more groups that alternately devise schedules. This approach is useful when one group creates resources required by another. For example, a university administration develops a schedule of classes, the students then submit their individual schedule requests, and the administration, after analyzing the requests, adds sections to some classes and deletes sections from others. The students then request changes to their schedules. In principle this cycle can continue for many iterations. This approach requires some incentive to cooperate and requires that each scheduler knows the global schedule and the state of available resources.

Any attempt to develop a universal scheduling methodology is doomed to failure because of the enormous diversity of scheduling domains. The variety of tasks, resources, constraints, and environments is virtually unlimited. The methodologies listed above are not applicable to all domains but must be selected based on the characteristics of the specific domain of interest.

Several other issues that are particularly relevant to distributed scheduling are briefly addressed in the remainder of this section. One of these is the requirement for revising a schedule,

also termed rescheduling [2]. Several factors can trigger a need to reschedule. A resource can become unavailable, making the current schedule unfeasible; a task can take longer than expected; or a user can change his or her requirements so as to impose a conflict, exhaust a resource needed by a later task, or delete an enabling task that creates a subsequently needed resource or state. In addition, rescheduling is desirable, although not required, whenever an opportunity arises to improve the schedule by adding previously unscheduled tasks or resequencing already scheduled tasks. This can happen, for example, when new resources become available or when a task completes early. Differences between scheduling and rescheduling include:

1. Rescheduling takes place in the context of an existing schedule that we may wish to disturb as little as possible;
2. Rescheduling must consider work in progress;
3. Rescheduling often must occur quickly, in contrast to the initial scheduling which may be performed in a more leisurely manner; and
4. Someone other than the original scheduler may perform the rescheduling.

An important issue for rescheduling in a distributed scheduling environment is the need to reduce communication requirements among schedulers to facilitate quick rescheduling. Since this may require a return to centralized scheduling, the rescheduler must have the appropriate information to make beneficial changes.

Another issue is that of database support for distributed scheduling. A distributed scheduling system requires many of the features of a distributed database management system. The system must merge separate databases of tasks, resources, constraints, and assignments into a single image while retaining the ability to display for individual schedulers only those portions of the database under their control. However, since each scheduler has a different view of the world (with different granularity levels, time scales, measures of goodness, types of constraints, etc.), the system must support different user languages and communicate with each scheduler in a natural and helpful way. As our software tools, such as COMPASS, address more diverse and complex problem domains, we will require a more comprehensive database language for describing scheduling problems.

Communication and coordination among schedulers in an important issue. NASA schedulers who impact one another may work at different centers, making communication difficult. The scheduling of Space Station Freedom will involve groups in several countries. An important research question concerns how frequently NASA schedulers communicate. Is the level of communication optimal? If it is below optimal, do schedulers fail to communicate because they do not perceive a need to communicate, or because they feel communication is too time consuming, or because they fear loss of control of their environment, or is there some other reason? If indepen-

dent scheduling is a human preferred approach, then it will be important to determine why this is true, how we can encourage people to cooperate, and how we can enhance cooperation while minimizing communication. The mechanisms for communication and coordination (the languages, database support, and interaction procedures) appear to be a critical aspect of distributed scheduling by human agents.

A final issue involves the introduction of expert system support for scheduling. Optimization heuristics have been envisioned for individual scheduling support; some of this support is already available on COMPASS. Expert system support for distributed scheduling would focus on communication and negotiation. An expert system that monitored the actions of all schedulers could infer when one scheduler needed to know of the actions of another. This technique could reduce communications requirements among human schedulers. Also, an expert system could search for instances in which two schedulers could trade resources or reschedule certain activities to their mutual advantage. Ultimately, we may wish to introduce artificially intelligent schedulers into a distributed scheduling system. The scheduling of certain domains, such as power generation, may be suitable for AI approaches. Once AI schedulers are developed for individual scheduling domains the natural next step would be to introduce them into human scheduling systems. This possibility raises questions regarding how artificial and human schedulers might best interact.

SAMPLE PROTOCOL: THE RED-BLUE PROTOCOL

The Red-Blue protocol has been devised to guide the interactions between schedulers at NASA/JSC and SpaceHab, Inc. of Huntsville, Alabama as they schedule STS-57, due to launch in April, 1993. The objective of this protocol is to facilitate the production of payload deployment and management schedules in the context of other orbiter/station operations. Based on our experience with scheduling this mission, the Red-Blue protocol will be enhanced and used as NASA's standard protocol for the distributed scheduling of shuttle (and, later, space station) operations.

The requirements for the Red-Blue protocol are similar to those discussed above. It should be easy to understand, use, and implement. Its use should be mechanical and unambiguous. It should work with a variety of scheduling software systems. It should support rescheduling. Its requirements for communication among schedulers, in terms of frequency and volume of communication, should be low. It should allow the creation of schedules in a timely manner. Finally, where there are conflicting objectives, the protocol should lead to the creation of schedules that provide a reasonable compromise between these objectives.

The Red-Blue protocol begins by dividing all of the activities into two groups, red and blue. The red activities can only be scheduled by the red scheduler, in this case, NASA. Likewise, the blue activities can only be scheduled by the blue scheduler, SpaceHab, Inc. Limits can be placed

on the volume of resources that can be used by the red and blue schedulers; for example, a scheduler might be limited to a maximum quantity of water during the mission or a maximum number of hours of an astronaut's time. Limits are not placed, however, on where in the schedule the resources may be used. In the case of NASA and SpaceHab, Inc., these limits have been established during contract negotiations. Any subsequent modifications or clarifications to these limits must be worked out by the schedulers and possibly their management.

The red scheduler produces the first schedule, placing red activities anywhere on the timeline, up to the limit of the red resource allocation. For example, the red scheduler would schedule basic activities such as course correction burns and astronaut sleep and meal times as well as mission specific activities such as payload deployment. Next the blue scheduler places blue activities in any available (white) space on the timeline, up to the limit of the blue resource allocation. Thereafter, only one scheduler may work on the timeline at a time. The scheduler who has authority to modify the schedule at any particular time is said to "hold the token." When the other scheduler has activities to schedule, that scheduler may request the token. The scheduler who holds the token may schedule or move his or her activities within any white space and within any space that he or she already occupies. The scheduler may not, however, move any activities of the other scheduler, or oversubscribe any resource.

If one scheduler wants to move an activity into the space owned by the other scheduler, the two schedulers can negotiate a set of changes that can then be produced by operations according to the basic protocol. While this protocol assumes that the parties are competitive (having differing and possibly conflicting goals), it also assumes that they are not antagonistic. Thus, the protocol assumes that the parties will cooperate whenever the result of such cooperation leaves neither party worse off.

A low communication procedure for asking the other party to move some of its activities is to allow a scheduler to oversubscribe resources (thereby producing a conflict between red and blue activities). The other scheduler, when he or she next holds the token, can leave the oversubscription (thereby delaying the resolution of the conflict), unschedule the offending activities of the other color, or accommodate his or her counterpart by moving some activities of his or her own color.

If more than two schedulers need to work cooperatively, then the Red-Blue protocol can be extended by devising a procedure for exchanging authority to operate on the schedule. A research question is to investigate the social and communications changes that occur as the number of scheduling groups rises.

Several implementation issues must yet be addressed. When activities must be rescheduled during a mission, does one party have the right to force the other to modify its schedule? For

example, if an activity runs long, can the other scheduler force termination of the activity? Also, electronic protocols must be developed for the exchange of schedule updates. These protocols must enforce the requirement that only the token-holder may modify the schedule and must ensure that all parties always agree on the composition of the current schedule. Thus, when one party refers to the current schedule or to the schedule as it existed two versions ago, the other party will know what is meant.

CONCLUSION

NASA has an unlimited variety of distributed scheduling problems. Competitive distributed scheduling problems arise when shared use of common resources interferes with the simultaneous achievement of multiple resources. We need to develop protocols that govern the process of building the schedule and govern how conflicting objectives are resolved. These protocols can only be developed and evaluated in the context of specific applications. This paper presents a simple, yet effective Red-Blue protocol to facilitate the production of payload schedules in the context of other orbiter/station operations.

REFERENCES

[1] Fox, Mark S., "Constraint-Directed Search," Ph.D. dissertation, Carnegie-Mellon University, 1983.

[2] Fox, Mark S. and Zweben, Monte, "Knowledge Based Scheduling," Tuturial MA2, presented at the Ninth National Conference on Artificial Intelligence, July 15, 1991.

[3] McDonnell Douglas Space Systems Co. "COMPASS 2.0 User's Manual," for NASA/Johnson Space Center Software Technology Branch, 1991.

[4] Ow, Peng Si, "Heuristic Knowledge and Search for Scheduling," Ph.D. dissertation, Carnegie-Mellon University, 1984.

[5] Salvador, Michael S., "Scheduling and Sequencing," in *Handbook of Operations Research*, Joseph J. Moder and Salah E. Elmaghraby (Eds.), Van Nostrand Reinhold, New York, 1978, pp. 268-300.

[6] Ullman, J. D., "NP-Complete Scheduling Problems," in *Journal of Computer and System Sciences*, Vol. 10, 1975, pp. 384-393.