N90-22304

328

MANAGING TEMPORAL RELATIONS

Daniel L. Britt
Amy L. Geoffroy
John R. Gohring
Martin Marietta Information Systems Group
P.O. Box 1260
Denver, CO 80201-1260

ABSTRACT

In this paper we will describe various temporal constraints on the execution of activities, and discuss their representation in the scheduling system MAESTRO* . Initial examples will be presented using a sample activity to be described. We will then expand upon those examples to include a second activity, and explore the types of temporal constraints that can obtain between two activities. Soft constraints, or preferences, in activity placement will be discussed. Multiple performances of activities will be considered, with respect to both hard and soft constraints. The primary methods used in MAESTRO to handle temporal constraints will be described as will certain aspects of contingency handling with respect to temporal constraints. We will conclude with a discussion of the overall approach, with indications of future directions for this research.

INTRODUCTION

In order to describe temporal constraint handling in the scheduling system MAESTRO, it will be helpful to first discuss in general what MAESTRO schedules, what a schedule is and how it's built.

The basic schedulable entity MAESTRO deals with is called an ACTIVITY. An activity is a set of actions which, when successfully completed, accomplishes some desired goal. We call these actions SUBTASKS, and specify that an activity is an ordered sequence of non-overlapping subtasks. For example, suppose we wish to perform a spectral analysis of a portion of the upper atmosphere using a satellite-born instrument. We could call this activity ATMOS. The sequence of subtasks which make up ATMOS are listed in table 1. We must power up the instrument, perform a self-test

* MAESTRO is a proprietary product of Martin Marietta Corporation.

on its electronics, calibrate it using a known light source, repoint it, collect the data we're interested in, and then put the instrument back in stand-by mode.

Table 1. Activity ATMOS.

Subtask	Name
1	Power Up
2	Self Test
3	Calibrate
4	Repoint
5	Collect Data
6	Power Down

In addition to descriptions of activities such as the one above, we need profiles of available resources used by the activities as well as profiles of ambient environmental conditions in order to schedule these activities. These profiles describe the state of each resource or condition as a function of time. Figure 1 shows an electrical power availability profile.

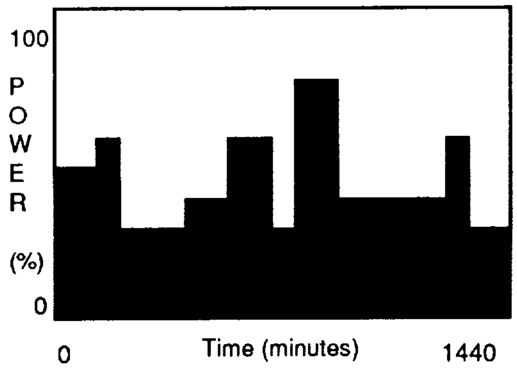

Figure 1. Electrical power availability as a function of time.

Given a set of activity descriptions and resource and conditions profiles, MAESTRO schedules by repeatedly executing what we call a select-place-update cycle. An activity is selected from among all activities requested to be scheduled. It is then placed on the schedule such that it can be executed as placed. Finally, its proposed use of satellite resources is noted on the appropriate resource availability profiles, and a calculation is performed to determine, within these new profiles, where on the schedule each remaining activity requested can be placed. MAESTRO will continue to execute these three steps, select, place and update, until there are no more activities requested to be scheduled which can be placed.

CONSTRAINTS ON SUBTASK EXECUTION WITHIN AN ACTIVITY

We will now begin to explore the various types of constraints which dictate where on the schedule our sample activity can be placed. Each of the subtasks making up the activity has certain resource and conditions requirements which must be met for the entire duration of the subtask in order for that subtask to execute. For example, the data collection subtask requires that the instrument be available, with enough electrical power to operate it in the proper mode, and that the instrument be pointed at the right area of the atmosphere, while not being pointed too near the sun. The instrument must also be in the right temperature range, and the platform must not be subject to too much vibration. Table 2 outlines these requirements for this subtask.

Table 2. Resources and conditions needed by ATMOS subtask 5.

Resource/Condition	Amount/Value
Instrument available	Yes
Power	400 watts
Target	Earth Limb
Sun Exclusion Angle	32 degrees
Temperature Min & Max	10 - 36 deg. C
Max Vibration	650 micro-g

Each of the resources and conditions listed above has an associated availability profile maintained by the scheduler. These can be used to generate lists of time windows during which a subtask can be running with respect to each requirement. The intersection of these windows determines when all resource and conditions requirements for the subtask are simultaneously met (see figure 2). In the MAESTRO system this is known as opportunity calculation. Since any number of lists of windows can be intersected, no limit is placed on the number of resources and conditions considered. Further, these lists of windows can come from any source, so a scientist can specify all those time windows during which he wants each subtask to be running. The user can restrict the performance of the whole activity to certain time windows as well.

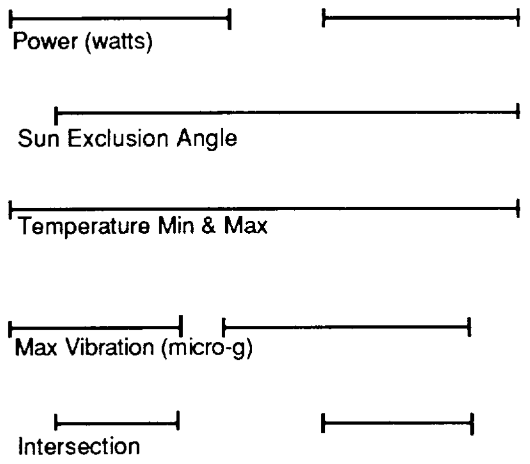

Figure 2. Time windows wherein a subtask can be "on" with respect each of four requirements, and their intersection.

The above calculation results in a clear picture of when each subtask in the activity can be running in isolation, but doesn't go far enough. Typically there are strict requirements on when each subtask can start and end relative to the placement of the others. The calibration subtask in the ATMOS activity, for example, must begin no later than 5 minutes after the self test subtask, which itself must immediately follow the power up subtask. In addition, each subtask has minimum and maximum durations specified by the scientist. The constraints listed above which determine the structure of the activity are constraints not on when each subtask can be running, but rather on when each can start and end. The two constraints used within an activity are the PRECEDES constraint and the FOLLOWS constraint. The end of the calibration subtask must *follow* its start by between 4 and 6, and must *precede* the start of the repointing subtask by at least 0 and no more than 10, for example. The complete list of these constraints is shown in Table 3.

Table 3. Subtask durations and delays for activity ATMOS.

Subtask	Duration		Delay	
	min	max	min	max
1	3	3	-	-
2	1	1	0	0
3	4	6	0	5
4	1	10	0	10
5	18	36	0	0
6	3	3	0	0

The precedes and follows constraints need not be applied only to adjacent subtask start and end points, but can constrain any two. Thus we can specify that the data collection subtask follow the calibration subtask by between X and Y. We can also limit the duration of the whole activity by placing precedes and follows constraints between the start of the first subtask and the end of the last.

CONSTRAINTS BETWEEN ACTIVITIES

Thus far we have considered only the constraints on a single activity, those which dictate its placement relative to resources, conditions and time windows, or its internal structure. Suppose, however, that in order to understand the data coming back from ATMOS in our example, the scientist needs to also get data from another instrument, in this case a solar spectrometer. This data must be taken at the same time that the ATMOS data is taken in order to correlate the results. We can describe the second activity, called SOLAR, much the way we did the first (see Table 4), but we also must specify the timing constraints between the two. We specify that the data collection subtask in the ATMOS activity must *start* and *end* at the same time as the data collection subtask in the SOLAR activity. These two new constraints, STARTS and ENDS, are similar to the

PRECEDES and FOLLOWS constraints referred to previously, and like those can have variable offsets. For example, we could specify that subtask 5 of ATMOS start between 2 and 5 minutes after the start of subtask 4 of SOLAR. Figure 3 shows the relationship between ATMOS and SOLAR.

Table 4. Activity Solar.

Subtask	Name
1	Power Up
2	Self Test
3	Repoint
4	Collect Data
5	Power Down

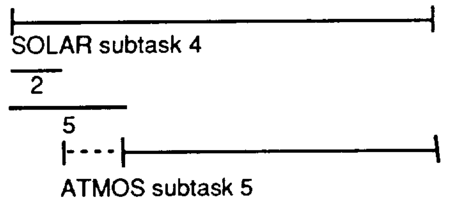

Figure 3. Temporal relationship between ATMOS subtask 5 and SOLAR subtask 4.

So far we have identified four temporal constraints - PRECEDES, FOLLOWS, STARTS, and ENDS. These dictate a relationship between a constrained entity and a constraining entity. The fifth subtask of ATMOS was the constrained entity in the example above. Specifically, the start of that subtask was constrained by the start of subtask 4 of SOLAR. All STARTS constraints will be of this nature. Similarly, all PRECEDES constraints will dictate a relationship between the end of the constrained entity and the start of the constraining entity. Table 5 shows the complete list of relationships specifiable with these four constraints.

Table 5. Relationships between constrained and constraining entities.

Constraint	Boundary of Constrained Entity	Boundary of Constraining Entity
PRECEDES	end	start
FOLLOWS	start	end
STARTS	start	start
ENDS	end	end

A fifth constraint type is necessary to fully specify possible relationships between constraining and constrained entities, the CONFLICTS constraint. Suppose we wish to specify that the calibration subtask of ATMOS will be disrupted if we try to perform the repointing subtask of SOLAR at the same time. We can indirectly represent this by causing SOLAR's subtask 3 to produce a condition, say vibration, which subtask 3 of ATMOS cannot tolerate, tracking vibration along with temperature and other conditions. More straightforwardly, we could specify that subtask 3 of ATMOS *conflicts* with subtask 3 of SOLAR. Like the others, this constraint can include variable offsets, allowing the constraining entity to block the constrained entity outside its duration.

It should be noted that the five constraint types listed above, with variable offsets, can singly or in combination express all of the relationships specified by Allen in his work on temporal constraint representation [Allen 1981]. We use a uniform interpretation of the meaning of positive and negative offsets on the constraints in MAESTRO such that a single algorithm can correctly propagate all of these constraints. An algorithm used for constraint propagation in scene understanding developed by Waltz [Winston 1984] has been modified for use in the scheduler. Given lists of time windows wherein each subtask can be running, it finds all and only those places on the schedule where each can start and end.

Notice that while the placement of ATMOS depends upon that of SOLAR, the reverse is not true. SOLAR can happen without regard to where or whether ATMOS is placed in order to achieve its own objectives. This is called a ONE-WAY or unidirectional constraint. If we wish to only perform SOLAR when it can support ATMOS, we can specify that it be a TWO-WAY or bidirectional constraint. A two-way constraint specifies a temporal relationship between two activities, and requires that neither can be scheduled without the other. Two-way constraints are more difficult to deal with than one-way constraints in MAESTRO. The select-place-update cycle described previously is designed to place a single activity, given complete knowledge (through opportunity calculation and temporal constraint propagation) of all possible placement options for that activity with respect to the current partial schedule. The existence of a two-way constraint precludes knowing all possible placements, since for each activity the position on the schedule of its constraining entity is not fixed. In MAESTRO we deal with this by placing more than a single activity on the schedule on each scheduling cycle. We calculate opportunity individually for each activity in a set of mutually related activities, then allow the constraint propagation algorithm to run on all activities in that set simultaneously. We call the set of mutually related activities a related set.

If the activities in the related set are independent of one another except for the temporal constraints putting them in the same set, the constraint propagation algorithm again finds all and only those places on the schedule where each subtask can start or end. However, if the activities share resource use or produce conditions which affect one another, this knowledge cannot be obtained with this algorithm. This allows the possibility that the placement of a related set will fail, necessitating backtracking. We have ruled out many placement possibilities that won't work and so can try various choices within those placements we think might work, and can apply various heuristics which are aimed at making sure each placement is significantly different from the last, but trial and error is involved if the activities share resources or have a producer-consumer relationship.

The use of related sets has been expanded in MAESTRO to include more than dealing with two-way temporal constraints. In our example involving ATMOS and SOLAR, we may have only a one-way constraint (ATMOS constrained by SOLAR), but may consider it much more important to schedule ATMOS than to schedule SOLAR. In this case we would like the two to be considered as both being important, and further would like the scheduler to only place SOLAR where it can support ATMOS. The related sets facility allows us to do this.

To this point we have considered only those situations wherein the placement of a subtask in one activity dictates where on the schedule a subtask in another activity can be placed. It is often desirable to specify an absolute time which constrains the start or end of a subtask in an activity. We also may wish to relate subtask placement to that of some event which will happen at various times but is not under control of the scheduler. Both these situations are handled in MAESTRO the same way we deal with one-way constraints between activities not in the same related set. Thus while constrained entities are always subtasks, constraining entities can be subtasks, events or absolute times. A constraint on an activity is represented as a constraint on the first or last subtask of that activity.

SOFT CONSTRAINTS

All of the constraints dealt with previously have been hard constraints, which must be satisfied in order for the constrained activity to execute. Another class of temporal constraints are soft constraints, or preferences. These guide the scheduler in placing activities on the schedule where it is most desirable, according to a scientist, platform manager, etc. Unlike hard constraints, however, these can be ignored if necessary to get things done. For example, it may be desirable to get data from ATMOS as near to noon, GMT, as possible, but not really necessary.

In MAESTRO, several types of soft constraints are representable. There are soft constraints which guide placement of a whole activity, called general preferences. Loading strategies are a type of general preference which guide placement of the activity with respect to the time period being scheduled. Front-loading, getting things done as early as possible, is a particularly attractive loading strategy in that activities scheduled earlier have a better chance of completing before something happens that might interfere with their completion. Various other loading preferences are supported. Other general preferences guide the structuring of the activity when there are variable durations for subtasks and/or variable delays between them. We can, for example, request maximizing durations and minimizing delays within the context of the loading strategy used for the activity being scheduled.

There are times when we wish to specify a preference which overrides these general preferences. We can, for example, ask the scheduler to place an activity where a particular subtask duration is maximized, regardless of where on the timeline that placement is found. This is called a specific preference, and is attended to in MAESTRO before any general preferences. Another type of specific preference guides the scheduler in placing activities either near to or far from other activities, events, or timepoints. Currently MAESTRO allows the specification of only one specific preference per activity, as the simultaneous satisfaction of two or more specific preferences is ambiguously defined.

Occasionally it happens that soft constraints on activity placement are at odds with the allowable placements as dictated by hard constraints, and these can interact in interesting ways. If a user requests that subtask 2 of ATMOS be placed as early as possible but the data collection subtask can only be placed late, the effect will be to stretch the activity out, maximizing delays between subtasks. In order to avoid this the scheduler under certain conditions will ignore general loading preferences and will intelligently order the application of general duration and delay preferences when one or more subtasks in the activity are highly constrained.

The approach taken in MAESTRO to scheduling typically yields "good" schedules, those which adhere to all hard constraints, pay attention to soft constraints when possible, and "get a lot done". It is sometimes desirable to ignore all preferences and just place activities randomly in an attempt to find a better schedule by generating several and choosing the best one, so MAESTRO has a random placement option. Using this option all hard constraints are still met, but a different schedule is generated each time the scheduler is run, allowing various activity placement combinations to be explored. Also, a user may wish to personally place some or all of the subtasks making up an activity, and this option is under implementation.

MULTIPLE PERFORMANCES

So far in this paper we have treated activities as if they were designed, scheduled, performed once and then forgotten. Typically, however, a user will want an activity to be performed many times. It can be the case that if an activity is not performed at least N times, it is not worth doing at all. Thus in MAESTRO a user can specify a minimum success criterion, a least number of performances acceptable to him. The scheduler uses these criteria in deciding which activity or related set to schedule next. The requirement to schedule multiple performances of activities makes scheduling more complex with respect to temporal constraints. If two temporally related activities each request several performances, and if there are variable offsets between them, it may be ambiguous which performances constrain which (see figure 4). MAESTRO maintains an interpretation of the relationships between performances such that constraints are never violated.

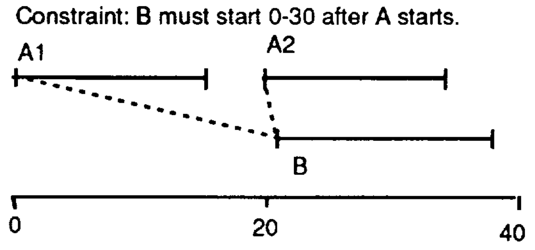

Figure 4. Multiple performances which may constrain others ambiguously.

A user may dictate that his experiment not be repeated more often than once every four hours, which introduces the idea of minimum performance separation. This is treated in MAESTRO somewhat like a one-way constraint. It is worthy of note that a negative performance separation, or overlap, is allowed by MAESTRO. A crewman performing an experiment in the lab module on Space Station Freedom may wish to begin preparation of a second sample before finishing the data analysis on the first, for example.

Typically when two activities are related by a temporal constraint it is required that one performance of the constraining activity be scheduled with one performance of the constrained activity. It may, however, be desirable to perform an activity once each third (nth) time that another is performed. This requirement is called a constraint arity. Facilities in MAESTRO for dealing with constraint arities other than one-to-one are not yet complete.

CONTINGENCY HANDLING

We have discussed a number of issues dealing with the generation of a schedule and the management of temporal relations involved. This scheduling is part of an ongoing operations environment wherein the assumptions upon which a completed schedule was based can change at any time, making the schedule invalid. It is preferable in most cases to alter the existing schedule rather than generating a whole new schedule for the time period encompassing the changes. Making changes to an existing schedule in response to changes in requirements, resource availabilities, etc., is known as contingency handling. One requirement levied on contingency handling processes is that they produce a modified schedule in which no temporal constraints are violated.

There are three aspects to contingency handling. One is simply scheduling; a late-arriving request to schedule an activity may only require that the activity be scheduled, with no other schedule changes. We have previously explored many aspects of temporal relations in scheduling. Another aspect is unscheduling, wherein a performance of an activity is removed from the schedule entirely in order to reduce resource usage, allow another activity to fit, or because a user no longer wishes to perform the activity. If an activity which constrains others is unscheduled, those others must be unscheduled as well.

The third aspect of contingency handling involves activities which have already begun to be executed but which cannot complete as scheduled. This may happen as a result of resource or conditions changes which become known only after the activity has begun, or in order to fit a high-priority activity on the schedule in response to a last-minute request. In this case it is desirable to make use of various characteristics of the activity to be interrupted and attempt to find a way to continue the activity. It may be possible to switch to usage of a resource other than that which was preempted by the contingency, leaving the activity structured the same as before. The subtask which was interrupted may be such that it can be continued after a short interruption with no ill effect, or it may be possible to begin at the start of that or an earlier subtask again after a pause, not beginning the whole activity again. Also, the rest of that subtask may not be necessary, as would be the case with a long data collection subtask during which more data was collected than required, allowing the activity to be continued by going immediately to the next subtask.

In each of these cases any temporal constraints between interrupted activities must be satisfied, possibly causing other activities to be interrupted, which may themselves allow restructuring. MAESTRO handles these situations by automatically generating activity descriptions which vary from the initial descriptions in ways allowed by the activity definition. These variant activities are called alternate models. It is assumed in MAESTRO that these alternates will satisfy the same temporal constraints as the initial model would, though in the real world that would not always be the case. Several versions of the MAESTRO scheduling system exist, and the facilities for handling these realtime schedule alterations in the ways explained above do not exist as described in all versions. For a more complete discussion of issues related to contingency handling, see Britt [1988a] and [1988b].

CONCLUSION

As is readily apparent from the preceding discussion, the handling of temporal constraints in scheduling is a formidable task. We have in this paper examined the ways in which the MAESTRO scheduling system deals with various types of constraints. These include resource and conditions constraints, windows during which subtasks can be running, constraints on the internal structure of activities, hard constraints between activities and other schedule entities, soft constraints or preferences in activity placement, and constraints between performances of the same activity. We briefly touched upon issues regarding contingency handling.

The approach taken by the designers of MAESTRO is to design solutions specifically for the problems in the domain, rather than trying to fit a predetermined solution paradigm to these problems. This results in a hybrid system making use of various methods and techniques as they are proven to work [Geoffroy 1990]. Proven techniques include object-oriented design, use of opportunity-calculation and constraint propagation algorithms to minimize backtracking (by getting optimal solutions to relevant subproblems at each step), use of user-derived heuristics such as front-loading, and a control structure that allows dealing with a related set of activities when appropriate. This approach to scheduling research is made feasible at least in part by use by the design team of a powerful and flexible software development environment supported by the Symbolics LISP Machine.

There is much yet to be done to complete the temporal constraint handling facilities in MAESTRO. Constraint arities other than one-to-one need to be dealt with more completely. The scheduler can be enhanced with the addition of smarter selection, placement and contingency heuristics. There are ways not yet implemented to deal with multiple specific preferences. User selection of the placement of individual subtasks is not complete, and the creation of alternate models of the same activity, which one version of MAESTRO performs, must be incorporated with the other capabilities previously described. This is by no means a complete list of scheduler enhancements that could be undertaken.

One effort that is anticipated to have enormous payoff, if it can be done, involves changes to the temporal constraint propagation algorithm itself. As explained above, backtracking is currently necessary in those cases where subtasks which can overlap also share use of constraining resources, as the scheduler cannot determine how those overlaps will affect resource availabilities given the variations possible in subtask placement. We hope soon to implement an algorithm similar to that which currently exists but with a major difference. The new algorithm will make use of information about possible subtask overlaps, and the increased resource use incurred, as well as the information we now use concerning when individual subtasks can be running, to find all and only those times when each of a group of possibly overlapping subtasks can start and end. The existing algorithm gives us this information for non-overlapping subtasks. Given this information about overlapping subtasks, the scheduler will be capable of scheduling sets of related activities without backtracking (trial and error). It will thereby be able to make full use of preferences in activity placement as well. Though it is not certain as yet that this calculation is possible, or computationally feasible, our experience with the current algorithm suggests that it is both. We intend that this and other new capabilities be installed in MAESTRO in the near future.

REFERENCES

Allen, J.F., (1983). "Maintaining Knowledge About Temporal Intervals." *Communications of the ACM*, 26(11), 832-843

Winston, P.H., (1984). *Artificial Intelligence*, pp 66-72. Reading, MA: Addison-Wesly

Britt, D.L., Geoffroy, A.L., & Gohring, J.R. (1988a). "Contingency Rescheduling of Spacecraft Operations." *Telematics and Informatics*, 5(3), 187-195.

Britt, D.L., Geoffroy, A.L., & Gohring, J.R. (1988b). "The Impact of the Utility Power System Concept on Spacecraft Activity Scheduling." *The Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference*, v. III.

Geoffroy, A.L., Gohring, J.R. & Britt, D.L. (1990). "The Role of Artificial Intelligence in Scheduling Systems." (In preparation).

Diagnosis/Monitoring