MISSION SCHEDULING FOR SPACECRAFT: THE DIARIES OF T-SCHED

B Drabble

University of Edinburgh, UK

Abstract

The aim of this paper is to describe the scheduling system T-SAT developed to automatically generate mission command sequences for the UOSAT-II satellite. At present the schedules are generated by hand which is a slow and potentially error-prone activity. While the exercise has shown our techniques to be successful with UOSAT-II, further investigations are required of applicability to more complex satellites such as ERS-1, before a complete picture can be gained. The paper provides a brief background to UOSAT-II, describes the design of the scheduling system and the representation of the constraints within it, discusses a typical scheduling problem and test results. Finally it suggests possible directions for future work.

1 Background to UoSAT-II

To investigate the uses of a store and forward communications satellite, UOSAT-II was launched into a Low Earth Orbit (LEO), in March 1984. Rather than remaining over one spot on the earth, polar orbiting satellites circle around the earth and pass over the poles. This circular orbit, combined with the earth's rotation, brings the satellite within radio contact of every point on the earth several times a day. A store and forward communication satellite can make use of this to act as an electronic "postman". Digital messages are received by the satellite when it is in contact with one ground station, stored in the satellite's computer memory, and then transmitted back to earth when the satellite later passes within range of the destination ground station. In addition to acting as an electronic "postman", UOSAT-II carries out a variety of other tasks. These activities include housekeeping tasks, e.g. telemetry; research work, e.g. VLSI chip reliability, and providing a variety of downlink information to schools and radio amateurs e.g. digitalker, whole orbit dumps.

2 Background to T-SCHED

The activation of any of these tasks within UOSAT-II is controlled from the "diary" programme. The diary is a weekly schedule which contains information on which tasks to activate, their start time, their duration of operation and their finish time. Associated with each task is a mission sequence which alters various switch settings on-board UOSAT-II e.g. to route information from data sources (experiments and internal devices) to the satellite's transmitters for forwarding to earth. The problem is however, that the individual mission sequences are not mutually independent i.e. two different mission sequences could require the same switch to be in different states if scheduled to occur together.

The problem of generating a schedule is further constrained by temporal considerations which again effect the possible ordering of mission sequences. As in most scheduling problems constraints can be specified as *hard* and *soft*. Hard constraints are those which cannot be relaxed *eg* "no data should be collected when the magnetorquers are fired". Soft constraints specify preferences which can be relaxed *eg* "Wednesday afternoons are dedicated to schools who prefer data transmitted on the 2m frequency band". Other constraints limit the duration of certain mission sequences *eg* "as the number of channels scanned in a whole orbit dump increases so the length of time they can be scanned for decreases".

The constraints described above are only two of thirty two which were identified by the designers of UOSAT-II. The number of constraints, their complexity and possible interactions mean that generating a diary by hand is a slow and possibly error-prone activity. To aid the human scheduler in the generation of such diaries T-SCHED was developed. The problems addressed within this project were as follows:

- To create a flexible time representation scheme which allows activities to be easily moved within the time line, should a problem arise.
- To represent the different types of constraints and discern how they could be relaxed within a general representation for scheduling problems.
- To provide the user with advice about possible times over which activities could be scheduled.
- To make the design flexible so that new constraints and mission sequences could be added.

3 The design of T-SCHED

T-SCHED is an interactive assistant to the human activity scheduler which provides help with the placing of an activity in a time-line and in the detection and correction of constraint violations. T-SCHED allows the user to enter activities in any order to the schedule and checks, as they are entered, that they do not violate any of the constraints specified. If a constraint conflict is detected then the user is informed of the conflict and of possible ways in which the conflict can be overcome. The options include moving the activity to a new time interval or changing some of the resources within the mission sequence.

A change in resource is restricted to changes of downlink frequency requested, but the system can be easily extended to deal with different internal storage devices and drivers.

Moving the mission sequence to another interval requires T-SCHED to search the time-line for an interval over which the conflict is removed and no new conflicts are introduced. This search is made difficult by the types of sequences which UOSAT-II can handle. A sequence can be specified as being *cyclic* (*i.e.* there are times when it is active and time when it is dormant).

Once T-SCHED has found a possible list of solutions these are presented to the human scheduler from which he or she must make a choice. If the human scheduler wishes to modify the mission sequence again at a later date, T-SCHED can support this as the time representation scheme allows for intervals to be easily moved within the time line. This can easily be achieved as all information entered by the human scheduler about a particular mission sequence is stored in a *knowledge frame*. The frame consists of a fixed number of *slots* into which information can be entered. The frame structure used in T-SHED for each mission sequence entered has the following format:-

```
Slot-1 The type of mission sequence, cyclic or one off

Slot-2 The function of the mission sequence

Slot-3 A data field to hold  variables associated with
       the function. For example a freqency band,
       the number of channels to be scanned, etc.

Slot-4 The start date of the misson sequence

Slot-5 The start time of the misson sequence

Slot-6 The end date of the misson sequence

Slot-7 The end time of the misson sequence
```

3.1 Representation of time in T-SCHED

The following two sections outline the motivation behind the time representation scheme used within T-SCHED and describes the actual scheme which was implemented.

3.1.1 Temporal considerations

One of the main aims of the T-SCHED project was to design a flexible and efficient time representation scheme. The problems in time representation which are specific to this problem are:

- The diary operates over the period of a week but the interaction between mission sequences can occur over minutes, thus the granularity of representation needed varies.

- The tasks themselves can be asserted as being *cyclic* or *one off*. [1].

  Cyclic Tasks :
  Cyclic tasks are entered into a *round robin* which is under the control of the real time multi-tasking operating system of UOSAT-II. The multi-tasking operating system gives each task a *time slice* during which it can execute and when this is complete it moves on to the next task in the round robin. A task will very often require more than one time slice and thus will remain in the round robin for several cycles. When the task has been completed it is swapped out of the round robin and *delayed* for a fixed period of a few seconds (10 to 240). When this time delay is completed the task is then re-entered into the round robin for re-execution. This process continues until the finish time associated with the task has been reached at which point it is swapped out of the round robin and never re-entered. Thus the number of cyclic tasks within the round robin varies over the period of the diary as new start times and finish times are reached. This means that we need to represent intervals occurring over minutes which cycle over days and whose interactions with other tasks may vary over time.

  One off Tasks :
  One off tasks as their name suggests are executed once within the schedule. They have a start and finish time as with a cyclic task *but* once the task has begun it is never delayed and thus remains constantly active until its finish time is reached. These tasks are relatively easy to represent but as they are constantly active they are quite likely to interfere with the resource requirements of other tasks. This means that a mechanism is required to detect interactions [1].

- The tasks themselves are not mutually independent, thus the time representation scheme needs to represent the assumptions under which the task can execute, i.e. to ensure all resources (switches, downlinks, etc.) will be available when required. If the scheduling of another task causes a violation of one or more of these assumptions *i.e.* a switch is required to be in alternate state, then this should be detected and reported to the scheduler/user.

3.1.2 The temporal knowledge base of T-SCHED

The time representation scheme used in T-SCHED consists of a single time-line in which the mission sequences are placed, and a truth maintenance scheme to record the dependencies which each mission sequence has on certain resources. In T-SCHED it is assumed that all resources are fixed and are always available if not assigned to another mission sequence or in violation of a constraint. This is unlike a typical planning system in which resource levels are variable and at certain times are unavailable.

When a new mission sequence is asserted, a check is made that the resources it requires are not in use by another block. This is easily achieved by associating with each previously asserted block a *justification* which indicates the resources which are required during the entire sequence. This justification can be described with the following statement:-

```
(derived before (start block1 (68 0)(69 1)(70 0)(4 0)))
(derived after  (end   block1 (68 0)(69 1)(70 0)(4 0)))
```

[2] The first statement states that before block1 can execute the switches 68, 69, 70 and 4 should have the settings 0, 1, 0 and 0 respectively. (When a new mission sequence is entered it is associated with a block number as an indexing mechanism.) The second statement states that the switches outlined should have the stated values and this should be true at the end of block1 execution. This allows the truth maintenance scheme to discern the difference between those switch settings which are only required to initiate a mission sequence and those required during the entire duration of a mission sequence. Each of these statements is given a *node* number and is used to build a *justification set* for the mission sequence.

```
(node-1
(derived before (start block1 (68 0)(69 1)(70 0)(4 0)))
(node-2
(derived after  (end   block1 (68 0)(69 1)(70 0)(4 0)))
(node-3
(sl (node-1 node-2) () block1)
```

The justification associated with node-3 can be broken down into three parts. The fields (node-1 node-2) and () are the *IN* list and *OUT* list respectively. The *IN* list states those nodes which must continue to be believed true for the justification to

[2]The word *derived* informs the dependency checker that the justification is not stored in the time line as a pattern but must be derived from information stored there

[1]The same task could be separately asserted as being cyclic as well as a one-off within the same diary providing no constraints were violated

remain true. The *OUT* list states those nodes which should not be believed for the justification to remain true. For example, I may believe the weather is fine because "the sun is shining" and it is "not raining". If either the sun stops shining or it begins to rain then I can no longer believe the weather is fine. The third field states the fact being justified *i.e.* that `block1` can be executed providing that the justifications described at `node-1` and `node-2` continue to be *IN*, i.e. true. If either `node-1` or `node-2` becomes *OUT*, i.e. false, then the justification for `node-3` becomes *OUT* and `block1` cannot be executed. This gives T-SCHED the ability to reason about the direct and *indirect* changes brought about by the assertion of a new block. For example, if another block required `node-3` to be *IN* and it fails then this too would be flagged as an error.

The start and end times of `block1` are not recorded in it's justification because a block may be moved several times during the schedule generation. Holding the time in one place makes updating much easier. The actual start and finish times for `block1` can easily be found from the time-line and its entries are as follows:

```
((block1 (start 9/10/89 12.00)(end 11/10/89 12.00))
 (block2 (start 11/10/89 9.00)(end 14/10/89 0.00)))
```

Thus using a simple calculation it is possible to determine if a newly asserted mission sequence will cause a violation with an existing mission sequence. If the new mission sequence causes no resource violations then it is checked against the temporal constraint described in the following section. If no further problems are found then the block is entered into the time-line and a justifications set is created for any resources which it may use.

3.2 Constraint handling in T-SCHED

The twin problems of representing and reasoning with constraints, and the temporal re-ordering of information within the time-line are at the heart of the T-SCHED scheduler. The constraints imposed by UOSAT-II are defined as being *hard* or *soft*. Hard constraints are those which cannot be relaxed, e.g. "do not collect data when the magnetorquers fire". Soft constraints represent preferences which can be relaxed, e.g. "Wednesday afternoons are dedicated to schools who prefer data transmitted in the 2m frequency band". Thus T-SCHED needs to represent the types of constraints present and the ways in which they can be *relaxed*. Constraint relaxation is a technique that allows a constraint to be "relaxed" in order to find a solution to an otherwise over constrained problem. The constraints which UOSAT-II imposes are represented in a constraint hierarchy as in ISIS and OPIS [2, 3]. The hierarchy includes information on how the constraints can be relaxed and the order in which this should be done. The constraints which T-SCHED can represent are as follows:

Duration :
This constraint indicates that the task has a maximum and/or minimum duration which should not be exceeded. For example, the duration of an whole orbit dump should not exceed (24 / Number of channels scanned) hours. This can be easily represented as an equation in which the length of the argument list in `slot-3` indicates the number of channels.

```
(constraint-1 (max duration (24 / length (slot-3))
                   min duration 0) hard whole orbit dump)
```

Not with :
This constraint indicates that certain tasks cannot be scheduled in parallel because of possible side effects. For example, if the magnetorquers are fired then no data can be collected

```
(constraint-2
   (magnetorquers-on whole orbit dump) hard)
```

Not on :
This constraint indicates that a task or a resource cannot be scheduled to occur at a particular time during the week. This time may be a single interval e.g. "between 9.00 and 12.00 on Monday morning" or alternatively it may be a cyclic constraint e.g. "all mornings between 9.00 and 12.00".

```
(constraint-3
   (single (9.00 12.00 Mon) telemetry) soft)
```

The above constraint specifies that if possible no telemetry should be sent on Monday between 9.00am and 12 noon.

```
(constraint-3
   (single (12.00 18.00 Wed) 2) soft)
```

The above constraint specifies that if possible the 2m downlink should not be used on Wednesday between 12 noon and 6.00pm. This refers to the fact that Wednesday afternoons are for school use who prefer data sent on the 2m downlink.

```
(constraint-3
   (all (9.00 12.00) whole_orbit_dump) hard)
```

The above constraint specifies that under no circumstances should a whole orbit dump be carried out on any morning between 9.00am and 12 noon.

At present these constraints are "hardwired" into the system in that they are read in by T-SCHED at the beginning of the session. However, T-SCHED could easily be modified to allow the user to incrementally input new constraints which could then be mapped on to the structures outlined above. (The constraints outlined here are in addition to those which do not allow a switch to be scheduled in different states simultaneously.)

Representation of the constraints is one aspect of the constraint handling problem. A second and more difficult problem is what to do when the current constraints do not allow a schedule to be generated. The solution which is most often used is to *relax* one or more of the constraints. However, this requires identifying *which* constraints can be relaxed, the *order* in which this should be done and the possible ways in which the constraint can be relaxed e.g. use a different resource, time slot, etc. The solution used in T-SCHED, as in most other AI systems is to specify the constraints as being *hard* and *soft*. Specifying a constraint as being *soft* allows constraints which are merely preferences to be identified. This is important because it is these which should be considered first in any relaxation process as they usually have a great deal of flexibility. A *hard* constraint specifies a constraint which usually has little or no flexibility for relaxation. With the constraints classified in such a way a hierarchy of constraints can be built up. The structure of the hierarchy is as follows: (level 1 is the top of the hierarchy and level 4 is the base.)

level 1 :
Hard constraints which involve time constraints. For example, do not collect whole orbit dumps on Wednesday mornings.

level 2 :
Hard constraints involving a resource but with no temporal bounds. For example, do not collect data when the magnetorquers are fired.

level 3 :
Soft constraints involving time restrictions. For example, the schools prefer the 2m to be used on Wednesday afternoons.

level 4 :
Soft constraints involving resources which can be substituted, but with no time restrictions. For example, a task may prefer the 70cm aerial over the 2m aerial.

As can be seen from the description the higher we are in the hierarchy the less flexibility there is to change the task within the schedule. The flexibility is specified by an additional field in which possible alternative resources are named. The constraint that no switch is scheduled to be in different states at the same time is handled within the time-line search routines and is not explicitly mentioned as a constraint.

If a constraint conflict is detected within T-SCHED by the assertion of a new control block or by an existing block being moved, then the user is informed of the options which are available to alleviate the problem. These are found from alternatives within the constraint hierarchy and from searching the time line for intervals over which the constraint's conflict is removed and no new constraints are introduced. Using this technique T-SCHED presents the user with a set of options geared to the particular constraint violated (of which there may be many) and thus does not require the user to create a "decision" rule base to provide guidance in overcoming the problem. However, if the human scheduler were able to define such control rules then T-SCHED could easily be extended to allow these to be integrated.

The options which can be suggested to solve a constraint violation are as follows:

Use an alternative resource :
This allows the user to substitute a resource with a similar one if such a resource exists and is allowable, e.g. substitute the 70cm for the 2m if the 2m is unavailable at the time requested.

Move block before :
This option allows the user to move the mission sequence (referred to as a block) to a time interval before the block which is causing the problem.

Move block after :
This option allows the user to move the mission sequence to a time interval after the block which is causing the problem.

Move block into cyclic loop :
This option allows the user to move the block with is at fault *into* the cyclic loop. This works because, the two blocks will now be kept apart by the round robin and thus cannot be active at the same time.

Move block out of cyclic loop :
This option is used to move a block out of a cyclic loop and into another cyclic loop or to make it a one-off action. This may be needed because the one-off action which is interfering with the cyclic action must be executed at a particular time and thus it is the cyclic action which must be moved.

4 Examples of T-SCHED in use

This section aims to show examples of T-SCHED and some of the schedules which it has successfully generated. The examples were generated from actual data supplied by the UOSAT-II team but to protect certain confidential information some parts have been changed.

The first example show the conflict between two mission sequences which require switch 68 to be in different states at the same point in time. As can be seen from the output, T-SCHED has correctly identified the problem and has supplied the user with a set of options from which to chose.

```
A conflict has been found with block 1

Type       : cyclic
Function   : telemetry
Switches   : [[4 0][0 1][69 0][68 0][67 0][66 0]
Start date : 12/10/89
Start time : 12.00
End date   : 18/10/89
End time   : 12.00

It involves the following switches
switch 68 is in conflict

Select option from Menu

1. Move One_off Before Cyclic loop
2. Move One_off After Cyclic loop
3. Move One_off Into Cyclic loop
4. Move block out of Cyclic loop

Type 1 to 4 :
```

In this particular example the option chosen was to move the one off digitalker task to a time interval before the cyclic telemetry task. T-SCHED provides the user with a list of time intervals during with the task could be executed. The human scheduler must then chose a task time interval which is entered through the usual input menu.

```
The time slots available are

Slot 1 now to 7 February 12.00 hrs

Slot 2 9 February 0 hrs to 12 February 12.00 hrs

Start date : 9/10/89
Start time : 12.00
End date   : 12/10/89
End time   : 12.00
```

The second set of examples shows how T-SCHED can detect and overcome constraint violations within a schedule. In the first example the user wishes to send data on the 2m aerial during a Wednesday afternoon, which is dedicated to schools who prefer data on the 2m aerial. As the choice of aerial was expressed as a preference it can be over-ruled by the user.

```
Input the information for the control block 1

Type            : cyclic
Function        : telemetry
Input the aerial : 2
Start date      : 21/10/89
Start time      : 12.00
End date        : 24/10/89
End time        : 14.00

*** Constraint violation ***

2m preferred: 14.00 and 18.00 Wednesday

Relax preference y/n : Y
```

The second example in this set shows how T-SCHED can handle the problem of there being no possible way to assign a new mission sequence to the schedule without invalidating one or more of the current constraints. In this example the attempt to carry out a whole orbit dump causes the violation of a *hard* constraint. This cannot be relaxed as with the previous example and no method has been supplied to relax the constraint relaxation. As a result the only action open to T-SCHED is to move the control block to a new time interval in which the violation is removed. The human scheduler is presented with a menu of possible options as in the previous example from which he or she must chose.

```
Input the information for the control block 22

Type                          : one off
Function                      : wod
The channels you wish to scan : 1 2 3 4 5 6 7
Start date                    : 21/10/89
Start time                    : 9.00
End date                      : 21/10/89
End time                      : 10.00

*** Constraint violation ***

No whole orbit dump allowed: 9.00 and 12.00

Constraint relaxation not possible

Select option from Menu

1. Move block before violated constraint
2. Move block after violated constraint

Type 1 to 2 :
```

As with the previous example the human scheduler must make a choice from the options given. In this particular example the option chosen is to move the block before the violated constraint. T-SCHED provides a list of valid intervals over which the problem is removed and no new ones are introduced. In the case of a whole orbit dump, this means periods every day between 9.00am and 12.00pm must be excluded from the list of choices giving us. The following fragmented output specifies this:

```
The time slots available are

Slot 1 now to 20 February 9.00

Slot 2 12.00hrs 20 to February 9.00 hrs 21 Feb

Slot 3 12.00hrs 21 to February 9.00 hrs 22 Feb

Slot 5 12.00hrs 22 to February 9.00 hrs 23 Feb

Slot 6 12.00hrs 23 to February 9.00 hrs 24 Feb

Slot 7 12.00hrs 24 Feb to schdule end


Start date :
Start time :
End date   :
End time   :
```

The next example shows that T-SCHED can add a mission sequence to a schedule without being instructed to do so by the user so as to avoid a foreseeable problem. In this example T-SCHED adds in a new mission sequence to turn off the 70cm aerial so as to avoid the possible situation in which both the 70cm and 2m aerials are active simultaneously.

```
Input the information for the control block 3

Type                          : one off
Function                      : wod
The channels you wish to scan : 1 2 3 4 5 6 7
Start date                    : 12/10/89
Start time                    : 12.00
End date                      : 12/10/89
End time                      : 14.00

extra block added: switch off the 70cm at block end
```

The addition of the extra block can be seen in the following schedule summary. The extra block (block 4) has been scheduled to occur immediately after the telemetry task has finished and it is a one off action.

```
Bk  Type     Function    Start    Time   End      Time
1   cyclic   telemetry   12/10/89 12.00  18/10/89 12.00
2   one_off  digitalke   9/10/89  12.00  12/10/89 12.00
3   cyclic   telemetry   18/10/89 12.00  21/10/89 12.00
4   one-off  70 off ***  21/10/89 12.00  21/10/89 12.01

blocks marked with *** added to avoid conflicts
```

5 Conclusions

T-SCHED has been successfully applied to the problem of generating mission sequences for the UOSAT-II satellite. To date T-SCHED has been able to reproduce the "diaries" created by human schedulers but it is hoped in the near future to have T-SCHED generate new sequences which can be uploaded to UOSAT-II.

The project itself has produced three main benefits:-

- We have gained a better understanding of the constraints which are operating in the domain of spacecraft command and control and the ways in which they can be represented.

- A prototype scheduling program has been produced which can aid the user to automatically generate mission sequences for UOSAT-II which are guaranteed to be free of interactions.
- A time representation has been produced which is flexible enough to handle events occurring over time periods of days while at the same time being able to handle interactions which may occur over seconds.

References

[1] B. Drabble. *Intelligent Execution Monitoring and Error Analysis in Planning Involving Processes.* PhD thesis, University of Aston in Birmingham, July 1988.

[2] M.S. Fox. *Constraint directed search: A case study of job shop scheduling.* PhD thesis, Carnegie Mellon University, 1983.

[3] S. Smith, M. Fox, and P.S. Ow. Constructing and maintaining detailed production plans: Investigations into the development of knowledge based factory scheduling systems. *A.I. Magazine*, 7(4), 1986.