Representation Of Scheduling Problems In Practice Of Mission Planning

Christoph Lenzen, Robert Rehm

DLR-Oberpfaffenhofen 82234 Wessling
christoph.lenzen@dlr.de

Abstract. When working on the theory of constraint satisfaction problems, one usually restricts to a finite set of equal variables, which can take values from a finite domain (see chapter 1, 1.1 in [1]). For many scheduling problems, this is not appropriate. For example: a schedule for the camera on a satellite shall be created. The visibilities of the targets usually won't appear as a multiple of a considerable large time unit. Besides the tasks will often not be equal but some of the tasks can build groups, which shall be scheduled as a unit (e.g. the uplink, which sends the command to the satellite, the datatake, i.e. the taking of the picture and the downlink, which sends the picture to the ground station belong together). The aim of this article is, to introduce an object language, which allows to model all sorts of constraints in an efficient way and which also enables the user, to define a structure, which can help the algorithm to be more efficient.

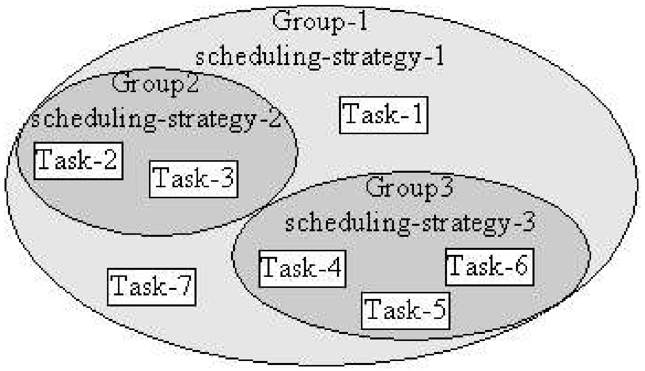
1 Structure of a scheduling problem

1.1 Tasks

A task is an activity, which can be assigned to any rational value at the timeline with any rational duration, so a task corresponds to a variable, whose domain is the square of the rational numbers. Therefore – regardless of complexity – an exhaustive search will never lead to a result. One either has to derive a granularity and an interval from the given data or one has to use algorithms, which don't need a finite domain.

1.2 Groups

A group is a collection of tasks and/or groups. No cycles are allowed (i.e. no group may contain itself, neither directly nor indirectly). Each group can be given its own scheduling algorithm. When starting the scheduler, it will consider only the tasks and groups, which don't belong to any group. When a group is chosen, the planning-algorithm of that group will be applied to schedule the elements of that group.
For example, our satellite, which has the camera to make a datatake, contains another experiment, called exp2. Then one can schedule all simultaneously, where both, the uplink – datatake – downlink groups and the exp2 groups are scheduled with their appropriate algorithm.



2 Constraint representation

There are three different possibilities to express constraints and a mechanism to combine them. A violation of these constraints is called a conflict.
The constraint types are:

2.1 Demand

This constraint type can cause a conflict when one task is scheduled and another one is not. There are two different types:

1. a task needs one or more other tasks to be scheduled, otherwise its occurrence on the timeline causes a conflict
2. n out of a given group of tasks have to be scheduled, otherwise any occurrence of any task of that group on the timeline causes a conflict

2.2 Time-Dependency

This constraint type can cause a conflict, when both included tasks are scheduled.

1. Start/end of task1 must be scheduled before start/end of task2
2. Scheduling times of task1 and task2 must overlap

etc.

2.3 Resource-Dependency

Resource

A resource is a function in time. For example, it can represent the battery-power of our satellite or it can model the times when some ground segment is visible for the camera of the satellite. There are different types of resources:

1. resources with upper-bound (another function in time), which cause a conflict, when the function raises above the upper-bound (e.g. memory, which is filled up by data from the camera – only a limited amount of memory is available, a conflict may occur)
2. resources with upper-bound, which force the function to stay below the upper-bound (e.g. battery power, which is refilled by solar panels – surplus energy is lost, but no conflict occurs)
3. resources without upper-bound (e.g. intensity of sunlight)

and the same for the lower-bound.

Resource-Dependency

A resource-dependency is a dependency of a certain task on a certain resource. There are different types of resource-dependencies:

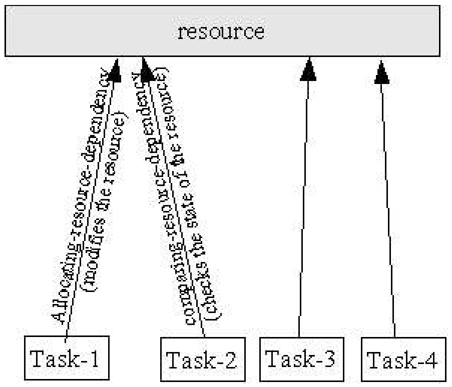
1. allocating-resource-dependency
   Modifies the function of a resource during the scheduling time of the task.
2. comparing-resource-dependency
   A comparing-resource-dependency may have an upper-bound and a lower-bound. This dependency causes a conflict, if the task is scheduled and if the resource function is out of these bounds during the scheduling times of the task. It does not modify the function of the resource.

Etc.

Resource-dependencies serve as a possibility to express constraints of tasks themselves (e.g. a datatake can only be scheduled when the target is visible) and to express constraints of many tasks among each other (e.g. only one downlink may be scheduled at the same time, since there exists only one antenna).

The resource-dependency is separated from the resource 'to allow one task to create an opportunity':

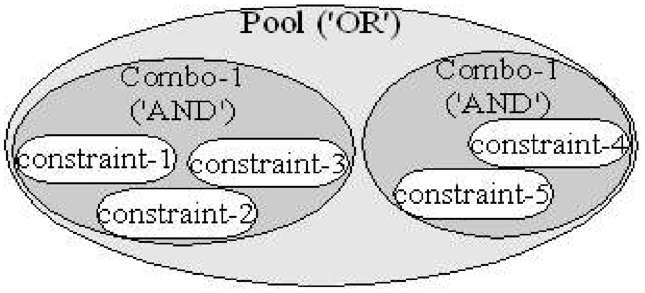
If the target of a datatake is not visible (represented by a comparing-resource-dependency, which is always violated), but it can be made visible by turning the satellite, a task 'turn-satellite' may modify the function of the resource, which describes the target visibility via an allocating-resource-dependency and so the target becomes also visible in the modelling.


resource
Allocating-resource-dependency (modifies the resource)
comparing-resource-dependency (checks the state of the resource)
Task-1
Task-2
Task-3
Task-4

2.4 Pools and Combos

A combo is a set of constraints. It is called 'satisfied', if all of its constraints are satisfied, i.e. if no constraint has a conflict and if all resource-dependencies, which modify a resource, are considered. Each combo must be part of exactly one pool.

A pool is a set of combos together with a number 'n'. All constraints, which belong to a combo, are ignored, instead the corresponding pool has to be considered. A pool is considered, if and only if exactly n of its combos are considered. It has a constraint, if one of the considered combos has a constraint.



3 Summary

The previously described object language is created to satisfy the demands of mission planning in practice. It allows to model structures of the planning-problem and it allows to model constraints in a natral and efficient way. The main difference between this modelling and the common constraint satisfaction modelling is, that there is a continuous timeline. Besides the grouping mechanism facilitates the mixing of algorithms. Both work well with simple heuristics like 'choose tasks/groups with high priority first', 'choose first possible time' and 'move conflicting tasks'. What has to be done next is, to adjust more elaborated algorithms, such that they can handle this modelling and to find ways to combine them properly.

References

[1] Edward Tsang, *Foundations of Constraint Satisfaction.* Academic Press, Harcourt Brace & Company, 1993