Modeling Constraints for Scheduling Problems

Mary Beth McMahon

The Boeing Company
13100 Space Center Blvd
Houston, TX 77059

Abstract

Commercial off-the-shelf products are often selected as a solution when faced with a new scheduling problem. These products perform poorly on finite capacity scheduling problems because they assume infinite resources and they do not model some of the harder constraints. This paper deals with our experience in deploying a finite capacity scheduler to several working environments. It describes the constraints that we encountered that caused us to modify the finite capacity scheduler in order to meet real world demands.

Introduction

When using some of the popular commercially available tools for scheduling, the current state-of-practice is based on constraints such as temporal constraints, precedence relationships and simple resource requirements. Commercial tools are often chosen because they are marketed as a pre-packaged solution and the advertisements show the public nice reports and friendly user interfaces that make it look easy to use the product to solve their problems. At first, the product seems to produce satisfactory results for the user. But as the user becomes more experienced with the product and the intricacies of the scheduling problem, the user runs into real-world constraints that cannot be modeled. This has been our experience for the past eight years when working with customers that were trying to improve their scheduling capabilities.

Some of the working environments in which we deployed a finite capacity scheduler included a testing lab where people load software and run simulations, a test and integration lab where people test hardware and software against different configurations, the SpaceHab Module, and the factory floor where people build aircraft. In adapting our software to meet their scheduling needs we had to model and schedule against some constraints that we had not thought about beforehand. Many of these constraints are generic enough that they are found in a number of domains in the space industry including assembly, mission planning, and mission operations.

The Underlying Scheduling Paradigm

Before discussing the constraints that need to be modeled, it is worthwhile to make sure there is a common understanding of the underlying scheduling paradigm that our finite capacity scheduler uses.

Each piece of work that needs to be done is modeled as a task. Each task knows about its own constraints: its earliest start time, the resources it needs, etc. The number and types of constraints are dependent on the application.

A Precedence Diagram (PD) can be used to model the precedence relationships between tasks. A PD is a diagram with boxes for tasks and arrows depicting precedence.

We define each resource, along with its initial availability, net usage, and list of attributes. An attribute is any characteristic by which a resource may be called upon, e.g. the skills associated with the resource (skill matrix). As tasks are scheduled and unscheduled, the net usage profile gets updated.

When a task gets scheduled it occupies a portion of the timeline and reserves the resources it requires. Subsequent tasks are scheduled around the tasks that are already scheduled. In our paradigm, scheduling a new task will not cause any already scheduled task to be effected in any way. Some schedulers shift tasks around to make room for a new task. Whether or not to perturb the existing schedule when adding tasks is an important consideration when selecting a scheduling tool.

In most of our scheduling problems, the goal of the scheduler is to come up with the shortest possible schedule while adhering to all specified constraints. There are numerous other optimization schemes, however for purposes of this paper minimizing makespan is a sufficient goal.

The Constraints

The following section describes the various constraints and behaviors that we needed to learn to model in order to tailor our system to handle real-world problems. The constraints included in this section are ones that most commercial schedulers fail to recognize. They are also constraints that can be found when scheduling space-related applications.

Constraint #1: Minimum/Maximum Duration

The duration of the task is a best guess at the length of time it will actually take to perform the work. The duration of a task may be more complex than just a singe number representing days and hours. In some cases the duration is flexible – usually there is a minimum duration and a maximum duration. The duration of the task should fall between the two. In the case of Boeing's Avionics Integration Lab, the scheduler gives each user the minimum duration requested and then backfills each task trying to stretch it to its maximum duration. This gives the user "as much time as possible" up to their maximum duration in the lab, while still ensuring that everyone gets in.

Constraint #2: Interruptible Tasks

Often tasks longer than a day must be interruptible or splittable. When splitting a task, rules about breaking it up must be captured. For example, if you do not specify the minimum time slice, or the shortest amount of time required to do productive work, then the scheduler may spread an hour long task out into 60 one-minute jobs. When splitting a task you may want to find out

- the minimum time slice,
- the maximum time slice,
- the minimum time between slices,
- the maximum time between slices, and
- the maximum time in which to accomplish the task (time space from start to finish including breaks).

Constraint #3: Tracking Tasks

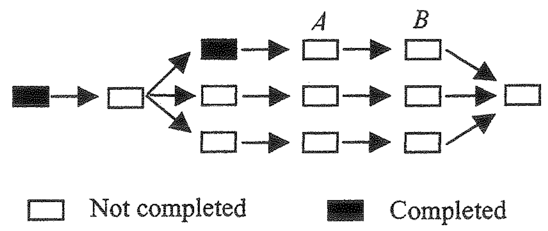
Once the schedule goes into production, the tasks must be updated as work gets completed and a new schedule must be published. Often tracking is done by interfacing the scheduling system to existing systems, such as a timekeeping system. In some cases email is used as the interface to the user in order to collect requests and amount of work completed.

In any case, the kinds of information that needs to be tracked is:

- the planned completion times,
- the actual completion times,
- the amount of work completed,
- the date/time a task is delayed until,
- the reason a task is delayed,
- the cancellation of a task and the reason, and
- the addition of new tasks.

Sometimes work is completed "out of sequence", that is, a task is completed before its predecessors. Although in theory this is not supposed to happen, in the real world it does and the scheduler needs to be able to respond appropriately. Sometimes once the task is completed, it allows its successors to be started. Other times the task's successors must still wait until its predecessors are complete. Sometimes the task completed out of sequence needs to be revisited after its predecessors are complete, usually for some minor work.

Work Completed Out of Sequence



When work is completed out of sequence, are the successors (*A* and *B*) enabled?

Figure 1. Work Completed Out of Sequence.

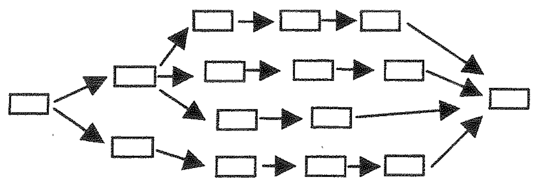
Constraint #4: Precedences and Preferences

The first step in deploying a scheduling system is to collect the data that needs to be scheduled. This includes all of the tasks, the resources, and the constraints. When collecting data it is often confusing whether the user is describing a constraint (a real requirement) or just a preference. A scheduler needs to distinguish between the two and model them both.

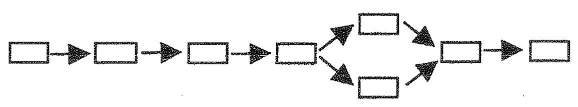
For example, when users create PDs showing the tasks to be done and their precedence relationships, we often hear that a group of tasks are linked by precedence because that is the order in which they perform the tasks. Even though there is not a requirement for doing them in that order, they assume the link because that is how it has always been done. Often there are good reasons to do the tasks in a particular order, but by modeling these as precedence relationships, we are unnecessarily introducing constraints and that will limit the solution space. However, if we don't model the order, we are losing valuable information - heuristics that a person has discovered about a good order for the tasks.

Modeling preferences and precedences introduces limiting assumptions to the scheduling engine which may compromise it's ability to come up with a better schedule than what is currently being worked. In extreme cases the user may dictate the order of all of the tasks, in which case there is only one solution, and the scheduling engine has lost all power to search for better solutions. In fact, the

Avoiding Superficial Precedence Constraints



A. The greater the parallelism the better the possibility of rearranging tasks to pack the schedule.



B. Superficial precedence constraints limit the scheduling engine's ability to find a better solution than what is currently being worked.

Figure 2. Avoiding Superficial Precedence Constraints

more parallel the tasks are, the greater the size of the solution space.

It is important to model both the real precedence requirements as precedence constraints and the preferred order as preferences. A scheduling engine may violate preferences in order to achieve a better value for an optimization function, but it may never violate precedence constraints.

Constraint #5: People Policies

One of the hardest things to model in scheduling is the subjectivity that goes into building a schedule where people are one of the driving resources. Not only are there union regulations governing working hours/overtime, but there are also "people policies" that managers employ to ensure that the schedule is humane.

In the case of the Systems Engineering Simulation Facility at Johnson Space Center, the workweek is around-the-clock, broken up into 4-hour "sessions." Their people policies include:

- A person can only work up to two sessions in one day and those sessions must be consecutive.
- Each person should have at least an eight hour break between non-consecutive scheduled sessions.
- If a person works more than one third shift then the third shift sessions should be either consecutive or at least two days apart.

In the Integration Lab in St Louis which is also scheduled 24 hours/day, 7 days/week, the people policies include:

- The maximum hours one can work in a "day" is 10 hours, but a day does not have to align itself with normal day to day time boundaries, (e.g. at midnight a new day starts) instead a day is defined by the times the user is assigned work.
- The minimum time between the completion of one day's activities and the start of a new day is twelve hours, eight of which must be on second shift.
- If the user requires specific hours, schedule them within the required hours or not at all. If the user does not require specific hours, but has preferences, let the user specify two sets of preferences. Attempt to schedule the task within the first set of preferences, if that fails attempt to schedule within the union of the first and second set of preferences.

Constraint #6: Preferred Resources

Another time users indicate preferences is when they describe the resources needed to complete a task. If there are multiple sets of equipment that can be used to perform the task, the user often has preferences on which set to use. The scheduling engine must 1) allow the user to specify alternate resources and 2) handle user preferences when selecting resources if applicable.

For example, when working on the assembly of aircraft, there are different categories of skilled laborers including a category called "ama" or "all around machinist/assembler". This person can do the work of a sheet metalist, an electrician, or a mechanic. In most tasks this person is an alternate resource that can be called upon to do the task. However, when scheduling we try to get the specialist first.

Constraint #7: Effect of Resources on Duration

The duration of the task often varies depending on the selection of resources. For example, one machine may mill a part in half the time it takes another machine to mill the part. Different skilled laborers perform the same job at different rates of time. In specifying alternate resources you may also need to specify how each resource effects the duration of the task.

Constraint #8: Limits of Physical Space

Often tasks are limited by the physical space surrounding the areas where the tasks are to be performed. For example, although there are more than enough resources at a given time to perform several tasks inside the cockpit of an F18, the space limits the number of workers to one or two at a time.

Similarly space is a consideration during the flight control play check. A person sits in the cockpit to run the tests. The areas around the aft portion of the wings, the rudders, and the stabilizers must be clear because there are 3000 lbs. of hydraulic pressure in the lines. If there were a single pinprick in the lines it could seriously injure a person if they happened to be working in any of those areas. So it is imperative to consider the space a task occupies when it is being performed.

It is very complex to model space in its true 3D form. Often a 2D representation or a mapping function can be used to adequately solve the problem. In our case we use "zones". The space around the aircraft is divided into predefined zones and each task can occupy one or more of the zones.

On the shop floors at the Cape where the space shuttles are refurbished, large pieces of equipment need to be moved around to prepare the area for work. If work in progress blocks a passageway and prevents a piece of equipment from being moved to the appropriate place, the tasks using this piece of equipment are delayed. So even though the equipment is available, the physical limitation of space prevents the equipment from being moved and the tasks from being performed. Modeling the geometry of the shop floor becomes an important factor when scheduling this case.

Affecting the Availability of Resources

We have run into three separate occasions when tasks have affected the availability of resources at times *other* than when they were scheduled. Traditionally, a task uses a resource during the time it is scheduled and returns it upon completion. The following examples illustrate instances that deviate from this behavior.

Production and Consumption of Resources

The completion of a task may result in the production of a resource used by a later task. Similarly, a task may consume a resource so that it is no longer available. This is the case in the missile systems area of Boeing. When manufacturing missiles, several tasks build subassemblies. Subsequent tasks use these subassemblies to build more complex subassemblies, and so forth. This building up of subassemblies and their use by subsequent tasks needs to be modeled. One way to do this is to model each type of subassembly as a resource. A task that creates a subassembly "produces" it. While a task that permanently uses a subassembly "consumes" it. The notion of the production and consumption of resources is a useful idea that is prevalent throughout space-related applications - especially in confined environments such as the space station, the shuttle, or SpaceHab, where resources, such as power, are used up and replenished.

The Same Resource for Multiple Tasks

Another occasion where resource availability is effected outside a single task is in the case of airplane assembly. In building large components of the aircraft, pieces are placed onto a jig where they can be riveted, sanded, and soldered. A number of different tasks can be performed while the pieces are attached to the jig. However, these tasks do not need to be performed in any specific order, nor do they have to be performed one right after the other. On many occasions, the labor will start some jig tasks, work on other tasks and then come back to the jig. Note that the jig is not available once the parts are mounted, but other resources such as equipment and labor used by the jig tasks are available to perform other tasks. Since there are a limited number of jigs, they are a highly constrained resource. Therefore the jig must be modeled as a resource that is unavailable once the parts are mounted, and available again once the jobs requiring the jig are complete. So the notion of several tasks securing a single resource needs to be modeled.

Intermittent Resource Changes

Lastly, there are instances when outages, malfunctions, and unplanned maintenance effect the availability of resources. When a resource becomes unavailable due to an unforeseen event, then the effect must be propagated throughout the schedule. A case where this has a significant impact is when aircraft assembly operations in St Louis change from two-shift to three-shift operations and vice-versa due to marketing or management directives. Sometimes this change is in effect for just a few weeks and sometimes it is in effect indefinitely. This kind of impact generally causes all tasks to be rescheduled in order to determine the effect on the schedule. The ability to update resource availability for specified periods of time needs to be modeled.

Overriding Constraints

Once we were able to model the constraints mentioned above, there were times when the manager wanted to override constraints. This is because the manager knew that work-arounds could be found - either people could share equipment, or they could put extra effort into a task to complete it despite the constraints. Sometimes the manager was willing to put up with an infeasible schedule to get a high priority job done, knowing that the conflicts could be dealt with sometime in the future

Almost every customer we have worked with has asked for this type of control. They want the ability to override certain constraints or schedule tasks at a given time, despite what the scheduling engine computes. They want feedback telling how the constraints are being violated because of their decisions. They often want to know the impact on the schedule to get ideas for how to work around the infeasible schedule and get back on track.

A scheduler should be able to enforce or relax constraints at the user's request. It should provide them with informative statistics about the goodness of the schedule and its validity or lack thereof when constraints are relaxed.

Conclusion

Commercial schedulers continue to be popular because they are readily available, easy to learn, and provide some very basic scheduling capabilities However, if the scheduling problem contains some of the advanced constraints described in this paper, there is no alternative but to search for a custom solution. Space operations are fraught with unusual scheduling constraints due to its highly constrained nature, its hazardous operations, and its lack of flexibility. Commercial schedulers which focus on project scheduling provide poor solutions for the types of problems we have encountered where the schedules need to reflect these more advanced constraints.