

A Constraint-Based Approach to Satellite Scheduling

Joseph C. Pemberton and Flavius Galiber, III
pemberto@psrw.com flavius@psrw.com

Pacific-Sierra Research
1400 Key Boulevard, Suite 700
Arlington, VA 22209

Abstract

Satellite scheduling, like all scheduling, is the problem of mapping tasks (observation, communication, downlink, control maneuvers, etc.) to resources (sensor satellites, relay satellites, ground stations, etc.). Through our work on satellite scheduling problems, we have encountered many different constraints that are particular to the satellite-scheduling domain. In this paper, we will introduce the satellite mission-operation scheduling problem, describing the problem constraints that are particular to satellite scheduling, and then present the constraint-based techniques that we have used to address these problems.

1 Introduction

In general, satellite scheduling is the problem of mapping tasks (observations, communications, downlinks, control maneuvers, etc.) to resources (sensor satellites, relay satellites, ground stations, etc.). Satellite scheduling problems differ from traditional scheduling problems in several ways. Perhaps the most obvious difference between satellite scheduling and traditional scheduling problems is the fact that some of the resources (i.e., the satellites) are orbiting the earth. This places an additional set of constraints on when a task can be executed. Satellite-scheduling problems oftentimes involve periodic tasks, variable length tasks, and tasks that can be preempted. In addition, satellite-scheduling problems are usually over constrained. Thus, satellite scheduling can usually be viewed as a constraint-optimization problem rather than a constraint-satisfaction problem.

Although the term satellite scheduling has been applied to many different aspects of a satellite's operation (e.g., design planning, launch control, lifecycle, etc.), we will focus on scheduling mission operations¹, namely the day-to-day activities of an operational satellite. Mission operation activities include payload operations (e.g., using a sensor on the satellite to collect data), bus operations (e.g., maintaining the health and status of the vehicle) and communications operations (e.g., transmitting data between satellites or to the ground and receiving information or commands from a ground station).

¹ This class of satellite-scheduling problems is often referred to as Satellite Mission Planning.

Our work has addressed three general types of satellite missions: broadcast, telecommunications, and remote sensing, although a single satellite may encompass more than one mission. Broadcast satellites (e.g., DirectTV, PrimeStar, etc.) are typically geostationary (i.e., they remain fixed above a single point on the earth's surface), and are used to broadcast information to a large number of receivers. The satellites are geostationary to allow for fixed position ground-based antennae. Telecommunication satellites (e.g., Iridium, Teledesic, TDRS, etc.) relay information back and forth between ground stations (e.g., mobile phones, communications antennae, etc.). Historically, telecommunications satellites have also been geostationary, but there has been a recent trend toward using constellations of medium- or low-earth-orbit satellites (e.g., Iridium). Telecommunication scheduling problems are made more interesting by the fact that communications between a satellite and the ground can pass through multiple relay satellites.

The mission of remote sensing satellites is to collect data and then transmit the data to earth. With the exception of weather satellites (e.g., GOES), remote sensing satellites (e.g., LandSAT, Hubble, etc.) are predominantly not geostationary, and consequently the ground area visible to the satellite changes over time. These visibility windows affect the availability of satellites to perform tasks. In addition to visibility windows, other events can directly impact the availability of a resource. For example, a satellite ground station may be unavailable for maintenance. Any satellite scheduling system must consider these events when building a schedule.

When it is not possible to satisfy all requests for satellite operations, a priority system is typically used to help choose which tasks to schedule. One possibility is for each task to be assigned a priority, and then a user defined objective function used to compare the relative merit of different partial schedules. In this case, satellite scheduling becomes an optimization problem.

This paper is organized as follows. The next section contains an overview of constraint-based scheduling, using a satellite scheduling problem as an example. This allows us to introduce some satellite scheduling terminology. Section 3 describes satellite scheduling in detail including a discussion of periodic tasks, preemptive tasks, and priority-based optimization. The next section describes our general approach to satellite scheduling, including some discussion of our approach to satellite specific scheduling challenges. The last two sections discuss related work and future work.

2 Constraint-Based Satellite Scheduling

Satellite scheduling involves assigning a resource (or set of resources), a start time, and duration to each task in a way that satisfies the constraints on the tasks and resources. We view this scheduling task as a class of Constraint Satisfaction Problems (CSPs). A CSP is typically defined as a set of variables, a set of values for the variables, and a set of constraints on the values that can be assigned to variables. Our basic approach can be described by two high-level steps: first we translate the problem into a CSP, and then we attempt to solve the CSP. In our approach, we model the satellite-scheduling problem

using four basic objects: tasks, resources, events and constraints. *Tasks* are the activities and operations to be performed. *Resources* are the people, satellites, sensors, communication channels, etc. that are required to accomplish tasks. *Events* are used to capture domain-specific occurrences that restrict when tasks can be scheduled (e.g., satellite visibility windows). *Constraints* are further restrictions on when tasks can be scheduled that are due to interactions with other tasks or to resource capacity and availability.

A task is scheduled by assigning it a resource, start time and duration, which satisfy all of the relevant constraints. Tasks that require more than one resource (e.g., a downlink task requires a satellite antenna and a ground station antenna) will need to have more than one resource assigned to it. More formally, a scheduled task is a tuple:

$$(\text{task}, \{\text{resource set}\}, \text{start time}, \text{duration})$$

In cases where the specific set of resources needed to perform a task are known *a priori*, then the scheduling problem is reduced to finding an execution time and duration for each task. When the tasks are fixed duration, then the scheduling problem is further reduced to finding an execution time for each task.

There are three types of satellite scheduling constraints: task constraints, resource constraints and event constraints. Task constraints are constraints between tasks. For example, a user may require two observation tasks to occur at the same time. Another example is a data download operation, which can only occur after the data collection task has been completed. In addition, there can be logical constraints between tasks, such as a requirement that one of two observation tasks be performed, but not both.

Resources also constrain the set of acceptable schedules. In the first place, resources have specific capabilities. For instance, a task to take an infrared image cannot be performed by a satellite that doesn't have an infrared sensor. In addition, a satellite that has a single infrared sensor can only take one picture at a time. A resource can either be required during the execution of a task (e.g., a sensor, communication channel, or downlink facility), or continue to be required past the completion of the task (e.g., fuel expended during maneuvers and on-board memory storage). For renewable resources, such as on-board memory, tasks (e.g., downlinking) can be used to replenish the resource capacity.

Other resources in addition to satellites must also be considered. Satellites can have multiple sensors with different capabilities. In this case, each sensor is a different resource. Satellites can also have multiple communications channels, each with a different bandwidth. Ground stations have personnel that are important resources as well.

Event constraints are used to describe time windows in which a task can be executed. In satellite scheduling, event constraints result from the fact that satellites must be able to see ground locations to perform a given task. Event constraints can also be specified by users (e.g., a satellite status check must occur between 23:00 and 24:00 each day).

Now that we have described the components of a satellite-scheduling problem, let's consider an example satellite scheduling task:

Satellite S must downlink to ground station G for 2 minutes between 23:05 and 23:27.

The task here is to perform a downlink activity, and the task duration is 2 minutes. The resources are satellite S and ground station G. The event window is from 23:05 to 23:27, which constrains when the task can be scheduled. In addition, this task can only be scheduled in a window when both satellite S and ground station G are visible and available.

3 Characteristics of Satellite Scheduling

In this section, we discuss several characteristics of satellite scheduling problems and briefly discuss their impact the scheduling task.

Event Windows

As mentioned in the previous section, the fact that some of the resources orbit the earth imposes an additional set of constraints on the scheduling task. In order for a satellite to perform a task (e.g., collect data, downlink, etc.), it must first be able to see the target. The times that a satellite is able to see its target are determined by the satellite's orbit characteristics. Detailed orbit propagation models are used to convert current satellite position and velocity information into a set of time-ordered positions of the satellite. The position of the satellite over time can be used to determine the visibility of the satellite to its target. The visibility start and stop times define the event windows for a given task-resource pair. If a task only requires a single resource, then the task event windows are equivalent to the event windows for the task-resource pair. If a task requires multiple resources (e.g., two satellites must both be in contact with a ground station at the same time), then the task event windows are the intersection of the event windows for each task-resource pair. Finally, if a task has alternate resources, then the task event windows are the union of the event windows for the task-resource pair. Once the task event windows are known, then they are used to constrain the task execution times.

Alternate Resources

In job-shop scheduling, it is commonplace that a task can be completed by more than one resource. This is also true for satellite scheduling problems with a twist. Because of the fact that no two satellites can share the same identical orbit and no two different ground stations can share the exact same ground position, alternate resources are not completely interchangeable. This is probably also true at some level of detail in job-shop scheduling (e.g., if the exact physical location of the machines impacts the actual execution time). The only cases where resources are interchangeable are for ground-based resources that are not position dependent (e.g., support staff, post-processing computers, etc.) and for geostationary satellites with nearly identical coverage areas.

A downlink operation may require the use of one of two relay satellites and a specific ground station. If satellite A is unavailable when it is in view of ground station G and G is available, we can't simply substitute satellite B because it may not be able to see G from its current position. This impacts the search as well. If there is an event constraint that keeps satellite B from satisfying a downlink task, then we don't want to backtrack over tasks for satellite B to search for a way to make it available for the downlink. In short, satellite B's availability is dictated by an event constraint that can't be changed by backtracking. Thus, alternate resources in satellite scheduling provide a different set of challenges than in traditional job-shop scheduling.

Periodic Tasks

Periodic tasks are common in satellite scheduling problems. Common periodic tasks include maintenance checks, downlinking collected data, and periodic collection over a specific region. Periodic downlink and maintenance tasks are tied to specific satellites, whereas periodic collection or communications support tasks are not necessarily tied to a specific satellite. Periodic tasks can be time or event based. An example of a time-based periodic task is:

Ground location L must be observed for 4 minutes every 12 hours.

An example of an event-based periodic task is:

Satellite S must downlink to ground station G for 2 minutes every time it can view the ground station.

These two types of periodic tasks are simply shorthand for a fixed set of regular tasks. Added difficulty occurs when the periodic task is specified relative to the execution of other tasks. Consider the following periodic task:

Ground location L must be observed periodically for 10 minutes with a maximum of 6 hours separation between observations.

In this example, the added difficulty comes from the fact that the number of observation tasks is not immediately available, but instead depends on how other tasks are scheduled.

Preemptive Tasks

Some satellite tasks can be preempted. An example of a preemptive task is:

Ground location L must be observed for a total of 30 minutes during the next 24 hours. Each observation must be at least 5 minutes in duration.

Preemptive tasks can be split up into a set of subtasks with shorter durations that sum up to the original duration, although the objective is to find a schedule with the fewest possible preemptive subtasks. Typically, the subtasks that make up a preemptive task can be scheduled on different resources. Preemptive tasks have a minimum duration for the subtasks, which gives an upper bound on the number of possible subtasks.

Having preemptive tasks complicates the scheduling problem in two ways. First, the scheduler doesn't know the optimal duration for a preemptive subtask (see variable length tasks below). The second complication is that the scheduler must decide how many subtasks to create.

Renewable Resources

Some satellite resources can be both consumed and renewed. The best examples of this are data storage and battery power. The problem that renewable resources pose is that the optimal number of resource renewal tasks may not be known a priori. Consider a satellite that can make two observations before it must downlink its data. If there are n observation tasks, then there must be at least $n/2$, and at most n , downlink tasks. The question is what is the optimal number of downlink tasks that are needed to satisfy the observation tasks.

To further illustrate this problem, consider the situation shown in Figure 1. The top line shows the visibility windows for performing observation tasks and the bottom line shows the times when the ground station is visible to the satellite (i.e., when a downlink can occur). If we assume that the satellite's memory is empty at time t_0 , then the first observation task will fill half of the available memory. At this point, the question is whether to schedule a downlink task during the first downlink window. Observe that if we perform a downlink during the first opportunity, we can then perform the two other observation tasks before the next downlink opportunity. Alternatively, if we wait until the satellite memory is full before performing a downlink, then the satellite will not be able to perform all three of the observation tasks. Performing too many downlink tasks is a waste of the downlink resources (satellite and ground station), whereas too few downlink tasks can result in an over-constrained scheduling problem (i.e., no feasible solution). Thus, the number optimal of downlinks needed to perform a set of observation tasks can depend on the way that the tasks are scheduled. Exploring these options increases the complexity of the scheduling problem.

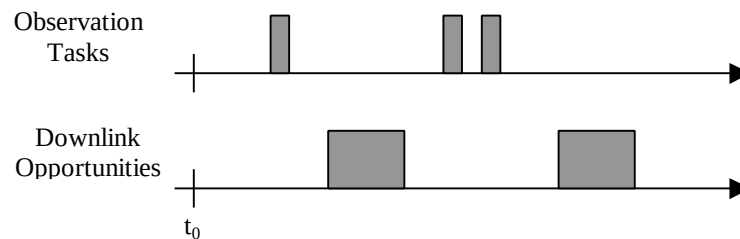


Figure 1: Downlink options.

Variable Length Tasks

When satellite-scheduling problems have tasks with variable duration, a satellite scheduling system must decide the duration for a task in addition to the start time and resource assignment. This complicates the scheduling problem because the scheduler must choose between scheduling more time for a task that has met its minimum duration and trying to schedule a different task. An example of a variable length task is:

Ground location L must be observed for a total of 30 to 90 minutes during the next 24 hours.

In these cases, a feasible schedule must at least satisfy the minimum duration, and schedules that allow for more observation time are preferred.

There are many other resources and tasks that are peculiar to satellite scheduling and that impact the complexity of the underlying scheduling problem. The examples presented in this paper are not exhaustive; rather they are intended to be an introduction to the satellite-scheduling problem domain.

4 Our Approach

Our general approach to solving satellite-scheduling problems is to first create a model of the tasks, resources, events, and constraints, and then search for a solution using heuristic-search with constraint propagation. We have implemented a system called GREAS (Generic Resource, Event, and Activity Scheduler [Pac97]), based on this general approach.

The modeling step involves translating the satellite-scheduling problem into a set of variables, which have ranges on their values, and a set of constraints on the acceptable values for the variables. Consider the following downlink task:

T_i : One satellite of $\{S_j\}$ must downlink to one ground station of $\{G_k\}$ for d minutes between times t_{begin} and t_{end} .

Our approach is to associate variables with each of the tasks and then the resources and execution start times become the values that can be assigned to the tasks. In this example, the task T_i is the variable. The job of the scheduler is to choose the resources and times that enable the task to be accomplished. For this example, the scheduler must choose a satellite and a ground station. Once a satellite and ground station are chosen, the scheduler's job reduces to finding a start time for the downlink task to be performed. The task description contains two explicit constraints on the value for the start time. The first is that the start time must be greater than or equal to t_{begin} . The second is that the start time must be less than or equal to $(t_{end} - d)$. In addition to the implicit constraints, there are also resource and event constraints. For example, the satellite and ground station must both be available (i.e., not servicing another task), and the satellite must be able to see the ground station.

Given a set of task variables with their associated value constraints, we make use of standard constraint-propagation techniques to help reduce the search space (e.g., [FS90], [Ilo98], and [SF90]). Consider the case where two tasks need to use the same satellite. Assume for this example that the satellite can only perform one task at a time, and that we have chosen an execution time for the first task. We can then propagate the "one task at a time" constraint by removing the execution time for the first task from the range of

possible execution times for the second task. Our current implementation of GREAS uses ILOG Solver/Scheduler [Ilo98] to perform the constraint propagation.

The last part of our approach is the search method. The general constraint-based approach to scheduling can be summarized as follows: choose a variable to instantiate, choose a value for the variable, and propagate the implications of the value selection on the remaining variables. This process is repeated until all variables have valid values or until one of the remaining variables has a null range (i.e., a deadend). When a deadend occurs, the search algorithm must backtrack by selecting a different value for one of the previously instantiated variables.

For any constraint-based scheduling problem, the underlying search method must address several questions [BPN95]. One is the order in which to instantiate variables (i.e., what order should the tasks be scheduled). For tasks that have multiple support variables (e.g., satellite, ground station, and execution time), there is the added question of the instantiation order for the support variables as well. Another question is how to order the selection of values for the variables (i.e., which value should be tried first, second, etc.). Over the past five years, we have developed a wide variety of problem-specific heuristics for ordering the variable selection and the choice of values.

Now that we have described our general approach to satellite scheduling, the remainder of this section will describe some steps we have taken to address the specific characteristics of satellite scheduling that were presented in the previous section.

Event Windows

Task event windows indicate the set of times in which a given task can be performed. We handle task event windows by creating an auxiliary task variable that indicates which event window will be chosen to satisfy the task. During the search process, the scheduler selects an event window and then attempts to schedule the task during that window. Currently, simple heuristics are used to order the selection of event windows.

Alternate Resources

The problem with alternate resources in satellite scheduling is that they are often not interchangeable. As mentioned in the previous section, this can lead to unnecessary backtracking. One approach to managing the selection of alternate resources is to create a separate task for each task-alternate resource event pair, and then further constrain these new tasks so that only one of the tasks can be scheduled. Although this preprocessing step creates a larger number of alternate tasks to consider, it also makes it easier for us to specify heuristics for choosing between the task options. The empirical performance of this approach has been reasonable on the problems we have considered to date.

Periodic tasks

We treat time-based periodic tasks as just a set of regular tasks. A task that occurs at 15:30 every day of a ten-day schedule is simply translated into ten different tasks.

Event-based periodic tasks require some preprocessing to determine their event constraints. Once the preprocessing is performed, these tasks can also be treated as regular event-constrained tasks.

Tasks whose periodicity is based on the execution of other tasks are a bit more problematic, and our scheduler does not currently handle them.

Preemptive Tasks

Consider the following preemptive task:

Use satellite S to perform an observation of location L for a total of 2 hours in at least 30-minute intervals.

This preemptive task can potentially be broken down into between one and four subtasks. Our current approach is to create all four subtasks with the following set of restrictions. The first task must be scheduled before the second task is scheduled, the second before the third, etc. The minimum execution times for each subtask are initially set to the minimum subtask duration. The start time of each subtask must begin after the end time of its preceding subtask. Finally, the sum of the subtask execution times must be greater than or equal to the minimum task duration and less than or equal to the total duration (i.e., 2 hours). The subtasks are then scheduled in order.

For some problems, the number of preemptive subtasks is too large to find a schedule in a reasonable amount of time. In these cases, we have adopted an iterative approach where the number of preemptive subtasks is initially small, and then increased, systematically, until a feasible schedule is found or until the time to optimize the schedule is exhausted.

Renewable Resources

As with preemptive and periodic tasks, the problem with renewable resources is determining how many resource-renewal tasks to perform. Since the optimal number of resource-renewal tasks depends on the scheduling decisions for the tasks that consume the resources, we have investigated several different heuristic approaches to this problem.

One method starts with the minimum number of resource-renewal tasks for each renewable resource and attempts to find a feasible schedule. If no feasible schedule is found, then the number of resource-renewal tasks is incremented for each renewable resource and the scheduling is repeated. This process is repeated until a feasible solution is found (if one exists) or until the available scheduling time is consumed.

A second method is to initially generate a “reasonable” number of resource-renewal tasks, and then search for a feasible solution that uses as few of the resource-renewal tasks as possible. A third approach is to require the user to specify the number of resource-renewal tasks that need to be scheduled, and then return a schedule that satisfies as many of the other tasks as possible. We have used all three of these approaches with some degree of satisfaction, and are considering other potential solutions to this problem.

Variable length tasks

Variable length tasks add an additional level of decision making because the actual duration of the task needs to be chosen. We have used different heuristics to address this problem. One approach is to find a feasible schedule using the minimum duration for all variable length tasks, and then modify the schedule by lengthening the duration of as many variable length tasks as possible. Clearly, this heuristic method does not guarantee an optimal schedule. Another approach is to try to find a feasible schedule using the maximum duration for all variable length tasks. If no feasible solution is found, then the duration of some variable length tasks is reduced and the search is repeated. Which task durations to reduce and by how much must be chosen heuristically (e.g., lowest priority first, longest duration first, etc.).

5 Related Work

The work presented in this paper is based on the experience that we have gained while developing and applying the GREAS Application Framework to satellite scheduling problems over the last five years. Our general approach is based on the constraint-based scheduling work of Fox et al. (e.g., [FS90] and [CS95]). Since our scheduling framework is built on top of ILOG Solver [Ilo98], we have directly benefitted from their constraint propagation methods.

Others have suggested different approaches to related observation-scheduling problems. In particular, several researchers at NASA Ames Research Center (e.g., [MJPL90], [DBS94] and [BME97]) have investigated different methods for scheduling observations from both ground-based and space-based telescopes.

6 Summary and Future Work

We have presented the satellite mission operations scheduling problem and discussed some problem features that are particular to satellite scheduling. We also discussed our general approach to satellite scheduling and our methods for handling these satellite-specific problem features. Our initial efforts have been incorporated into our commercial satellite scheduling tool (GREAS). We are currently developing metrics for measuring the relative performance our models and search algorithms, and we are actively investigating new models and search techniques to improve the performance of our scheduler.

In the future, we will continue to investigate different methods for modeling the satellite scheduling problem domain. We are also developing and evaluating different search methods. We also plan to continue exploring the peculiarities of satellite scheduling through a more formal analysis of the problem domain. Finally, we are in the process of generating a set of test problems and objective functions to support empirical studies.

Acknowledgments

This work has benefited from discussions with members of the Resource Optimization group at PSR and from discussions with several of our customers.

References

- [BME97] John L. Bresina, Robert A. Morris and William R. Edgington. Optimizing observation scheduling objectives. In proceedings, *1997 NASA Workshop on Planning and Scheduling for Space*, 1997.
- [BPN95] Phillipe Baptiste, Claude Le Pape, and Wim Nuijten. Incorporating efficient operations research algorithms in constraint-based scheduling. In proceedings *First International Joint Workshop on Artificial Intelligence and Operations Research*, Timberline Lodge, Oregon, 1995.
- [CS95] Cheng-Chung Cheng and Stephen F. Smith. Applying constraint satisfaction techniques to job shop scheduling. Technical Report CMU-RI-TR-95-03, The Robotics Institute, Carnegie Mellon University, January 1995.
- [DBS94] Mark Drummond, John Bresina and Keith Swanson. Just-in-case scheduling. In proceedings, *National Conference on Artificial Intelligence (AAAI-94)*. American Association for Artificial Intelligence, 1994.
- [DW91] Thomas Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, San Francisco, California, 1991.
- [FS90] Mark. S. Fox and Katia P. Sycara. Overview of CORTES: A constraint based approach to production planning, scheduling and control. In proceedings, *Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head, South Carolina, May 1990.
- [Ilo98] ILOG Corporation, Gentilly Cedex, France. *ILOG Optimization Suite: White Paper*, April 1998.
- [Le92] Wiley J. Larson and James R. Wertz (editors). *Space Mission Analysis and Design*. Kluwer Academic Publishers, 2nd edition, 1992.
- [MJPL90] Steve Minton, Mark D. Johnston, Andrew B. Philips and Philip Laird. Solving large-scale constraint-satisfaction and scheduling problems using a heuristic repair method. In proceedings, *National Conference on Artificial Intelligence (AAAI-90)*. American Association for Artificial Intelligence, 1990.
- [OS97] Angelo Oddi and Stephen F. Smith. Stochastic procedures for generating feasible schedules. In proceedings, *Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. American Association for Artificial Intelligence, July 1997.

- [Pac97] Pacific-Sierra Research, Arlington, Virginia. *GREAS Application Framework Reference, Version 4.0*, November 1997.
- [Pap95] Claude Le Pape. Three mechanisms for managing resource constraints in a library for constraint-based scheduling. In proceedings, *INRIA/IEEE Conference on Emerging Technologies and Factory Automation*, Paris, France, 1995.
- [SF90] Norman Sadeh and Mark S. Fox. Variable and value ordering heuristics for activity-based job-shop scheduling. In proceedings, *Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head, South Carolina, May 1990.
- [ZF94] Monte Zweben and Mark Fox. *Intelligent Scheduling*. Morgan Kaufmann, 1994.