Incremental Scheduling Engines – Cost Savings through Automation

John Jaap & Shaun Phillips

Mission Operations Laboratory
Marshall Space Flight Center
National Aeronautics and Space Administration
EO50, MSFC, AL 35812
John.Jaap@nasa.gov Shaun.Phillips@nasa.gov

Abstract

As humankind embarks on longer space missions farther from home, the requirements and environments for scheduling the activities performed on these missions are changing. As we begin to prepare for these missions it is appropriate to evaluate the merits and applicability of the different types of scheduling engines. Scheduling engines temporally arrange tasks onto a timeline so that all constraints and objectives are met and resources are not overbooked. Scheduling engines used to schedule space missions fall into three general categories: batch, mixed-initiative, and incremental. This paper, presents an assessment of the engine types, a discussion of the impact of human exploration of the moon and Mars on planning and scheduling, and the applicability of the different types of scheduling engines. This paper will pursue the hypothesis that incremental scheduling engines may have a place in the new environment; they have the potential to reduce cost, to improve the satisfaction of those who execute or benefit from a particular timeline (the customers), and to allow astronauts to plan their own tasks and those of their companion robots.

Introduction

The National Aeronautics and Space Administration (NASA) is charting a bold new course into the cosmos, a journey that will take humans back to the Moon, and eventually to Mars and beyond. Currently, operations is the most expensive part of many space missions; the cost of operating the *International Space Station (ISS)* is in excess of one billion dollars per year. Current planning and scheduling methods need to be replaced with automated methods. In addition, every avenue should be taken which can reduce the stress and tedium endured by astronauts. A mission to Mars is expected to take about two years; and, during much of the mission, light-time delays (typically 10 to 15 minutes) will negate voice conversations with the ground. One good way to address these human factors issues is to give the astronauts control over their daily schedule. The current "job-jar" paradigm used on the *ISS* only allows the astronauts to select additional optional tasks which use limited resources. True astronaut control would allow them to schedule or re-schedule most tasks. For some foreseeable contingencies, complete crew autonomy in planning and scheduling may be required.

The scheduling systems of the future must be able to handle both the complexity of the tasks and procedures (to ensure a valid schedule) and the flexibilities of the procedures and the equipment (to effectively utilize available resources).

Scheduling Overview

Planning and scheduling software temporally arranges tasks onto a timeline so that all constraints and objectives are met and resources are not overbooked. The scheduling unit or scheduling request addressed by the scheduling software is an "operations sequence" containing multiple tasks and the temporal relationships between the tasks.

> Example: Dinner is a scheduling unit, which includes tasks that prepare dinner, eat dinner, and cleanup. All tasks must be done and they have temporal relationships; each follows the other.

In the space activity domain, temporal relationships such as sequential, overlap, during, and avoid are common. Additionally the operations sequence can have parallel paths, repetitions, and other arrangements. In the domain of human space flight, operations sequences are often networks, with embedded sub-networks. To emphasize the complexity of the problem, the term *task networks* will be used for the remainder of this paper. A task network with only one task and no temporal relationships is the trivial case of a network, and is the simplest scheduling unit.

The tasks of a task network use resources. In the dinner example, each task would use several resources (power, microwave oven, water, food stock, waste disposal, etc.). Tasks might also have condition requirements which constrain the scheduling to happen before, during or after a certain condition, such as in daylight.

The computer representation of a scheduling unit and its components is called a model. The quantitative and

associative values in a model are expressed either as rules or as fields in a dynamic hierarchy of forms.

The planning system's core logic, or scheduling engine, must understand the models and must temporally arrange multiple complex task networks to generate a valid schedule. In the space activity domain scheduling engines use algorithmic, heuristic, artificial intelligence, and human-assisted techniques to solve the space scheduling problem.

Classes of Scheduling Engines

Scheduling engines are generally considered to be "batch," "incremental," or "mixed-initiative" based on how they handle multiple scheduling requests. See Figure 1.

Batch Scheduling Engines

Batch engines accept a batch of independent scheduling requests and put them on a timeline by assigning the start and stop times of each task. The tasks to be scheduled are often associated only by the use of the same resources. Batch engines search for an optimum or near-optimum timeline based on analytical, heuristic, algorithmic and/or artificial intelligence techniques. The search methods used by batch engines must simultaneously meet the requirements of many tasks and many independent temporal networks. This implementation places a limitation on the modeling schema and is computationally intense.

Schedule-repair engines are a special case of batch engines; the batch of requests fed to the engine is the tasks already on a timeline, but having constraint violations.

The primary attribute of batch schedulers is the ability to optimize the use of resources and maximize value of the timeline. In fact, of the three classes of scheduling engines, only batch engines can produce near-optimum timelines. For this reason they are the engine of choice for unmanned space probes and similar missions.

Incremental Scheduling Engines

Incremental engines add each scheduling request to a timeline without adjusting the times of already-scheduled tasks and without introducing constraint violations or resource overbooking. The core logic of an incremental engine is usually some form of a greedy algorithm; that is it makes choices based only on scheduling the current request. Like batch engines, these engines may use analytical, heuristic, algorithmic and/or artificial intelligence techniques. The logic required to handle the temporal networks of a single scheduling request is less difficult to develop than the logic required by batch engines which need to handle all the temporal networks in the timeline. As a result, the models presented to an incremental engine can be more complex and can capture more of the requirements or can capture the requirements more accurately.

As an incremental scheduling engine schedules a request, it behaves like a batch engine with respect to the multiple tasks of the scheduling requests. However these tasks always have temporal relationships to each other and may share the same resources. All tasks scheduled by previous scheduling requests are locked and the residual resource profiles are treated as initial resource profiles for the current request.

Incremental engines do not provide global optimization. However, a "Monte-Carlo" technique is available to overcome this limitation. Multiple schedules can be produced by submitting the scheduling request in different orders; a figure of merit can be assigned to each schedule; and the best schedule chosen as the solution. Heuristics and analytical logic can be applied to find a submission order which gives good results.

One novel use for incremental engines arises when multiple users are building a single timeline; the engine allows each user to add tasks and be sure that subsequent action by other users will not change the times of those tasks. Later this paper presents an in-depth discussion to two possible uses of incremental scheduling engines that exploit this feature.

Mixed Initiative Scheduling

Mixed-initiative scheduling refers to building a timeline using a timeline editor; i.e., it is a manual process. Mixed initiative is used when the user knows requirements that are not described in the requests, the scheduling engine is weak, only a few new requests are to be added to the timeline, or the user wants to control the results. If the user moves already-scheduled tasks, mixed initiative has characteristics of a batch scheduler; if the user doesn't move already-scheduled tasks, then mixed initiative has the characteristics of an incremental scheduler.

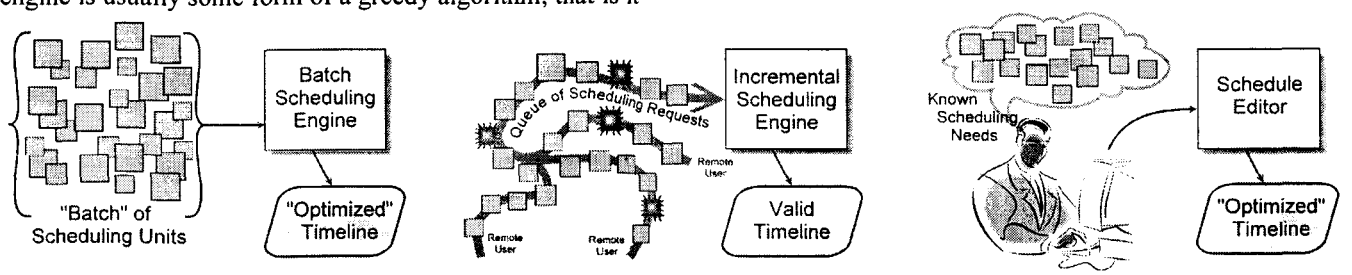

Figure 1 - Classes of Scheduling Engines

Mixed-initiative schedulers usually have code to help the user avoid violating constraints and often allow the user to override constraint limits. In the batch flavor, if the models are complete, the editor might invoke iterative-repair logic to move other tasks and eliminate constraint violation introduced by a manual edit. In the incremental flavor, the editor might invoke an incremental scheduler to make slight adjustments to the user's input; this feature is called "snap-to." Additionally, the editor might use an incremental engine to suggest times where tasks can be placed without introducing constraint violations.

Mixed-initiative scheduling does not automatically provide global optimization. However if the human user is an expert and the problem is straight-forward, global optimization might be achieved.

Currently, all human space missions are scheduled using mixed initiative.

Reducing Costs

Customer participation in *ISS* operations scheduling is an example of reducing costs by using an incremental scheduling engine. This example assumes the use of an incremental engine accessed via the web by the users of the timeline – each user would schedule his tasks without fear that the tasks will be moved.

In this example, use of the scheduling engine is distributed to the actual timeline customers (those who benefit by the execution of the timeline, such as scientists, technicians, systems operators, or others who have a stake in the mission). Customers access a central installation of the scheduling system using remote access technology. An overview of the operations concept is shown in Figure 2.

The example concept has a weekly scheduling phase that produces the timeline to be executed during the second week after it is produced. During any week, three actions are occurring: the week after next is being scheduled, next week is being prepared and up-linked, and the current week is being executed. The preparation phase is closely linked to what equipment is on board, which is linked to crew change-out or the arrival of a re-supply ship. In ISS nomenclature an "expedition" is a period of time that is punctuated by a crew change-out; nominally, an expedition is 90 days. The preparation phase for an expedition precedes the start of the expedition; the scheduling phase begins two weeks before the expedition starts and continues for the duration of the expedition.

During the preparation phase, the customers define what equipment they need and/or will supply and how it is to be used. The cadre creates the equipment mode models based on the customers' needs and the cadre's own knowledge of how the equipment is installed in the ISS. Models may later be updated by the cadre as needed. Additionally, a high-level plan for the expedition is generated based on customer input, programmatic goals and constraints, and various agreements with the partners.

Based on the expedition plan produced during the preparation phase, and other information, the cadre would generate daily allocations per payload for the week to be scheduled. The allocations are not usage profiles but are total usage limits of each resource during the planning week. Once the system is initialized with all the resource constraints, the customers use the incremental scheduler to produce a timeline. As always, producing a good timeline requires attempting to schedule a model, rejecting unacceptable results, tweaking the models, and trying again. The incremental scheduler places the customer in the middle of this important iteration loop. No one knows the payload requirements or what is desired in the timeline better than the customer, and no one can produce a timeline as good as the one the customer can produce.

After the customer has completed the scheduling process, the timeline is delivered to the cadre for timeline verification. Verification consists of checking that safety and other criteria are not violated. The cadre will also visually inspect the timeline and the models.

After the timeline is verified, it is passed to the integration function where it is integrated with timelines from other ISS partners. Simultaneously, it is "published" so that the customers can review the timeline. If a customer wants to have the schedule changed, an execution change request is written and submitted to the execution team. Since the customer just produced the schedule (of his payload), it is unlikely that changes will be required.

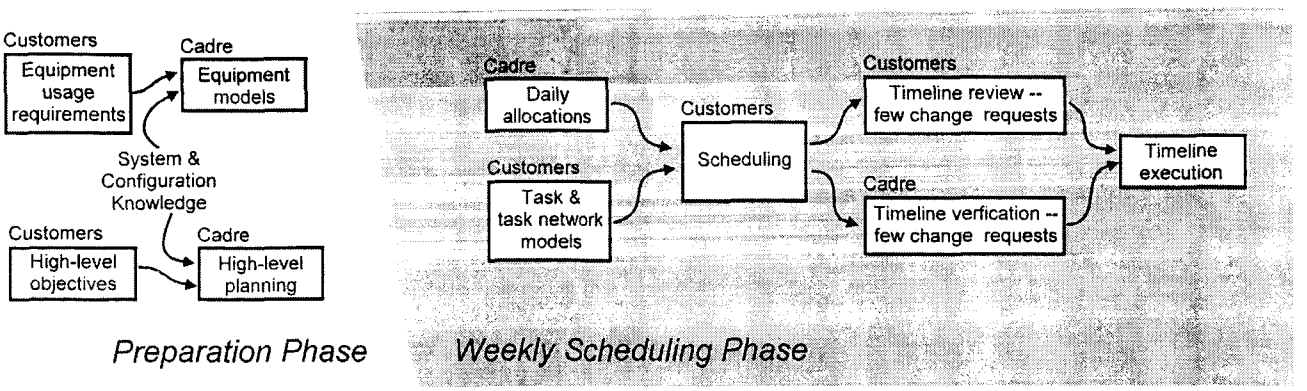

Figure 2 - Customer Participation Operations Concept

Astronaut Participation

Astronaut participation will be important on long-duration human missions. On short flights like those of the Space Shuttle and intermediate duration missions like an ISS expedition, the activities of the crew are primarily scheduled by the ground controllers. Lack of crew planning autonomy has been a topic of discussion for decades, and there is anecdotal consensus among astronauts that crew autonomy is a good way to mitigate the stress of long-duration missions. Incremental schedulers have the potential to allow true astronaut participation in planning their own daily schedule. One possible implementation is to co-locate the scheduling engine with the astronauts, and ground support personnel (controllers, scientists, and others) would remotely access the extraterrestrial engine. See Figure 3.

An earth-based engine is used to build baseline models and timelines. The space-based engine is used to update the timeline; these updates can be made by the astronauts or by earth-based personnel using the remote access capabilities of the incremental scheduler. This concept provides the astronauts with a complete set of up-to-date planning information, and allows them to make any additions they desire to the currently executing timeline.

The level of astronaut participation in the scheduling process will be dictated by necessity (e.g., responding to real-time events) as well as by their personal preferences. In effect, the concept provides an infrastructure which allows multiple parties (astronauts and ground personnel) to simultaneously contribute to the development/ maintenance of a single timeline.

Once the planning information is within the scheduling system at the extraterrestrial site, it will be available for use by the onboard astronauts. From a local console, they will be able to view/inspect their timelines, make timeline changes (by deleting and rescheduling), schedule additional "job jar" type tasks via an interface to the incremental scheduling engine, and even edit the modeled tasks (e.g., change a specified task duration). An interface via a personal data assistant could provide access to the habitat

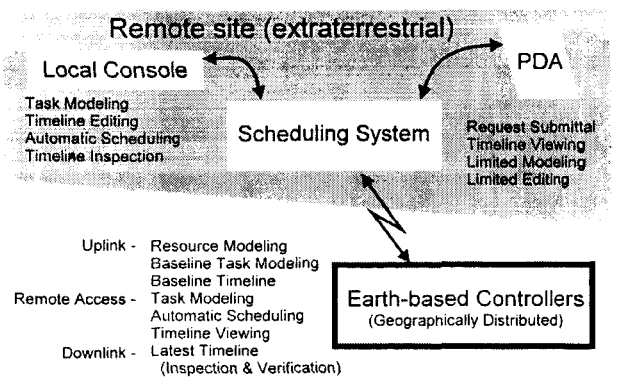

Figure 3 - Crew-Participation Operations Concept

installation of the scheduling system and allow adding to the timeline during outside excursions.

Earth-based controllers can also remotely access the extraterrestrial scheduling system to inspect/verify the most current timeline information or to contribute timeline changes. To preserve precious crew time, it is envisioned that most extensive re-planning efforts will be performed by the earth-based controllers, except in those cases where communications outages or delays preclude a timely ground response to a real-time event. The earth-based controllers may also perform timeline edits at the crew's request.

Conclusion

NASA is charting a bold new course to explore the cosmos beginning with humans returning to the Moon and anticipating a human visit to Mars. Without significant advances in operations concepts, these missions will be more expensive than necessary. Additionally, there is a compelling need for astronaut autonomy to address human factor issues and contingency issues introduced by light-time delays.

Of the three classes of scheduling engines (batch, incremental, and mixed initiative), incremental engines offer significant promise to reduce cost and provide substantial astronaut participation. The attributes of incremental scheduling engines which enable cost reduction and astronaut participation are:

- The logic of the engine can handle more complete/complex models because it schedules only one model at a time. More complete models mean that less information is maintained outside of the model and less mixed-initiative scheduling is needed.
- The scheduling cadre does not need to be experts on the objective of the temporal networks being scheduled. They only need to provide knowledge of the hardware and the vehicle/habitat systems.
- Astronauts do not need to be experts on the tasks within the models. They can submit any pre-defined model to the scheduling engine and, since the model contains all the requirements, produce a valid schedule.
- Users can add to the timeline without danger of modifying what is already scheduled. For example, ground controllers are assured that astronaut additions to the timeline do not impact critical tasks.
- A user who is scheduling a given model does not need to know anything about other models or what is already on the timeline.
- The independent handling of each scheduling request allows simultaneous remote access to the scheduling engine by multiple users. Thus, scheduling can be distributed to those who have the best knowledge and vested interest in producing a good timeline.