Incremental Scheduling Engines for Human Exploration of the Cosmos

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. One way to reduce the cost of humankind's journey into the cosmos is to develop new methods of planning and scheduling and new software to support those methods. The current approach of scheduling human missions manually needs to be replaced by automated approaches.

In addition, every avenue should be taken which can reduce the stress and tedium of extended space missions far from home. 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. This job jar is prepared on the ground and uplinked daily. True astronaut control would allow them to schedule or re-schedule most tasks. For some foreseeable contingencies, complete crew autonomy in planning and scheduling their daily tasks may be required.

The search for cost reduction and astronaut autonomy must take into account that the equipment on the journey will be the most complex hardware ever developed, the information sought will be at the cutting edge of human endeavor, and the procedures will be intricate and exacting. Scheduling will be made more difficult by a scarcity of resources. The scheduling system 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. It does not make sense to do only one or two of these tasks; i.e., to schedule a partial sequence.

The tasks also have an order or temporal relationship; each follows the other.

The example above is a simple operations sequence. In the space activity domain, temporal relationships such as sequential, overlap, during, and avoid are common. Classically, there are thirteen temporal relationships possible between two tasks (Allen, 1983). 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, of tasks. 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.) in differing and specific quantities. Tasks might also have condition requirements which constrain the scheduling to happen before, during or after a certain condition, such as in daylight. Condition requirements are a form of temporal requirements, but are not labeled as such to avoid confusion with temporal relationships between the tasks.

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 core logic of the planning and scheduling system must understand the models and must temporally arrange multiple complex task networks to generate a valid schedule. This logic is commonly called a scheduling engine. In the space activity domain scheduling is NP-hard (has no analytical solution), therefore scheduling engines use algorithmic, heuristic, artificial intelligence, and human-assisted techniques to solve the space scheduling problem.

Scheduling engines fall into three general categories: (1) batch, which operate on multiple models simultaneously, (2) incremental, which operate on one model at a time, and (3) mixed-initiative, which is a human and software working together. In the space domain, most scheduling systems use more than one of class of scheduling engine.

Classes of Scheduling Engine

Scheduling engines are generally divided into three classes based on how they handle multiple scheduling requests. The classes – batch, incremental, and mixed-initiative – are discussed below.

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 requirement places a limitation on the modeling schema. However, unless all requirements are modeled, the resulting timeline has little chance of being valid. Additionally, scheduling a large number of requests may require considerable computer resources.

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 and/or resource violations. Conflicts occur when the initial timeline was produced with relaxed constraints (limited overbooking allowed), or when a batch engine is presented with simplified, or high-level, models and the detailed models are used by the schedule repair engine. These engines use algorithmic, heuristic, and/or artificial intelligence methods to resolve the conflicts. Since the repair of one violation sometimes introduces other violations, an iterative approach is implemented. Good iterative repair engines have logic to avoid local minimums and find the overall optimum solution and logic to lock selected tasks so that they will not be moved by the repair process. A example of an implementation of an iterative repair engine is the Automated Scheduling and Planning Environment (ASPEN) (Rabideau, 1999) developed by the Jet Propulsion Laboratory.

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. A great deal of work has been done to optimize these engines with the goal of developing autonomous scheduling systems to be installed on spacecrafts.

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 (Cormen, 2001); 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 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, one 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. This approach is commonly called a Monte Carlo solution. Heuristics and analytical logic can be applied to get a submission order which gives good results. An example of an incremental scheduling engine with a Monte Carlo submitter is the Experiment Scheduling Program (ESP) (Davis, 1988) developed by the Marshall Space Flight Center.

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. Section 3 of 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.

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. The Mars Exploration Rover mission uses mixed-initiative scheduling as implemented in the Mixed Initiative Activity Planning Generator (MAPGEN) (Bresina, 2004) developed by Ames Research Center in concert with the Jet Propulsion Laboratory.

Reducing Costs

Incremental schedulers have the potential to reduce cost. Customer participation in ISS operations scheduling is used to illustrate one way an incremental scheduling engine might reduce cost. 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.

Introduction

In this example, use of the scheduling engine is distributed to the actual customers of the timeline. The customers are those who benefit by the execution of the timeline; they may be 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. The typical steps required to build a timeline are listed below.

- **Flight hardware integration.** The customer community provides descriptions and requirements of the flight hardware as needed by their tasks. Customer supplied hardware is integrated into the flight systems.
- **Equipment models.** A scheduling cadre (specialists who assist the customers) builds equipment mode models for flight hardware. These mode models define the resources and constraints of the hardware in all of its operating modes. Note that the scheduling cadre does not need to understand the tasks which will use the equipment.
- **Task models.** Using remote access, customers build their own task models. They have the best knowledge of their requirements and are in the best position to build their models. They do not need to know the details of the resource requirements of the equipment; that data is in the equipment models built by the scheduling cadre. For example, a customer requesting a voice link does not need to request a communication link or specify bandwidth consumption; this information is in the voice-link mode model.
- **Task network models.** Using remote access, customers build their own task network models. Their objectives

drive the temporal relationships between the tasks to be done. They know the best task ordering (sequenced, overlapped, during, etc.) to accomplish their goals. A task network model is the scheduling unit that an incremental scheduling engine schedules.

- **Submitting.** Using remote access, a customer submits task network models (the scheduling units) to the scheduling system. The scheduling system adds each request to the queue of requests for the incremental scheduling engine.
- **Results.** The results of each request are displayed at the user's display device when the request is operated on by the scheduling engine. If the scheduling engine scheduled the request, the results are displayed for the customers; the customer can then accept or reject (delete) the results. If the engine cannot schedule the request, an explanation is displayed.
- **Re-submitting.** For rejected and failed request, the customer can modify the request and resubmit.
- **Timeline verification.** Since an incremental scheduler is designed to never overbook resources or violate other constraints, the scheduling cadre only ensures that the timeline meets criteria such as safety. The customers have developed the timeline to meet their objectives; the cadre does not have the knowledge or the need to improve the timeline. Verifying the mode models is not needed since only the cadre can change them.

Using an incremental scheduler to distribute timeline generation to the customers can provide better customer satisfaction and reduce the size of the scheduling cadre.

Customer participation operations concept

This concept was first proposed in a paper by Jaap and Muery (Jaap, 2000). An overview of the operations concept is shown in Figure 1.

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.

Evaluation

This operations concept for ISS utilization does not provide an optimum schedule. This is no worse than the current operations concept based on mixed-initiative which also does not provide an optimum schedule. This operations concept does have the potential to provide a better schedule to the customers while reducing cost. It can provide a better schedule because those who have the best knowledge of objectives are the actual builders of the timeline. It can reduce cost by reducing the size of the scheduling cadre – they no longer need to be experts in the objectives being scheduled.

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 (Compton, 1983; Hagopian, 1998; Sherman, 1994), 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. The astronaut participation example is focused on the exploration era when there will be astronauts on the Moon, Mars, and/or on long-duration cruises.

Introduction

In this example, the scheduling engine would be co-located with the astronauts, and ground personnel (controllers, scientists, and others) would remotely access the extraterrestrial engine. The typical steps needed to allow astronaut participation in building the timeline are listed below.

- **Baseline timeline.** Ground personnel build a timeline using a local engine. The steps to build this timeline are similar the steps used to build the timeline with customer participation as discussed above.
- **Timeline Uplink.** The timeline and models are uplinked to the extraterrestrial engine.
- **Astronaut additions.** Astronauts add to the timeline as they desire using the extraterrestrial engine. Using an incremental scheduler ensures that tasks added by the astronauts do not change or conflict with previously scheduled tasks.
- **Re-verification.** When time allows, the timeline modifications are downlinked to the ground and verified.

Locating the scheduling engine in space will provide the astronauts with the ability to manage the schedule and will enable more autonomous crew/vehicle operations. The astronauts will be able to make a real-time schedule change and get immediate feedback that the change is feasible. When the astronauts are far from home with significant light-time delays (up to 40 minutes round-trip to Mars), and the return home time measured in months, astronaut autonomy will enable the safest, most reliable, and efficient approach to exploring the cosmos.

Example operations concept

This concept was recently proposed by Jaap and Maxwell (Jaap, 2005). An overview of this concept is depicted in Figure 2.

The astronaut participation concept assumes an installation of the scheduling engine on earth and another in space. The 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 remote 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 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.

Evaluation

This operations concept for the exploration of the cosmos does not provide an optimum schedule. However, it provides significant participation by the astronauts in the development of the timeline. Astronaut participation can ameliorate some of the human factor issues anticipated for missions exceeding two years in length and where all voice communication with the earth is excluded by the light-time delay. Additionally, the example concept will allow full autonomy if needed; for example, in case of loss of communication for a significant duration.

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.

The examples provided in this paper illustrate the use of incremental scheduling engines to reduce costs and provide astronaut participation and possibly astronaut autonomy. However, significant shifts in operations concepts will be needed to take full advantage of the attributes of incremental engines. NASA managers are challenged to consider the necessary paradigm shifts.

REFERENCES

Allen, J. F.; "Maintaining Knowledge about Temporal Intervals," Communications of the ACM, Vol 26, No 11. , November 1983.

Bresina, J.; Jónsson, A.; Morris, P.; Rajan, K.; "Mixed-Initiative Constraint-Based Activity Planning for Mars Exploration Rovers," Fourth International Workshop on Planning and Scheduling For Space – IWPSS04 Proceedings, Darmstadt, Germany, June, 2004.

Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; and Stein, C.; Introduction to Algorithms, 2nd Edition, Sec. 16, The MIT Press, Cambridge, Massachusetts, 2001.

Compton, W.D., and Benson, C.D., Living and Working in Space: A History of Skylab, National Aeronautics and Space Administration, Washington, D.C., 1983.

Davis, E.; Jaap, J.; "The Scheduling Techniques of ESP," Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88), Dayton, Ohio, July1988.

Hagopian, J., Maxwell, T.G., and Nahay, E., "NASA/MIR Phase 1: A Lesson in Long Duration Mission Planning and Operations," in proceedings of Space Ops 98, the Fifth International Symposium on Space Mission Operations and Ground Data Systems, Tokyo, Japan, June 1998.

Jaap, J.; & Maxwell, T.; "Enabling New Operations Concepts for Lunar and Mars Exploration," Space Technology & Applications International Forum (STAIF-2005) Proceedings, Albuquerque, New Mexico, 2005.

Jaap, J.; & Muery, K.; "Putting ROSE to Work: A Proposed Application of a Request-Oriented Scheduling Engine for Space Station Operations," Sixth International Conference on Space Operations (SpaceOps 2000) Proceedings, Toulouse, France, June 2000.

Rabideau G., Knight R., Chien S., Fukunaga A., & Givindjee A.; "Iterative Repair Planning for Spacecraft Operations Using the ASPEN System;" in proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i SAIRAS), Noordwijk, The Netherlands, June, 1999.

Sherman, J.D., and Maxwell, T.G., "Decentralizing Astronaut Flight Scheduling on the Space Station," Business Case Journal, Volume 2, Issue 1, Summer 1994, pp. 95-103.