ON THE NEED FOR DYNAMIC SCHEDULING OF IMAGING SATELLITES

J. C. Pemberton[a], L. G. Greenwald[b]

[a] Veridian, 14150 Newbrook Drive, Suite 300, Chantilly, VA 20151 – joseph.pemberton@veridian.com
[b] Department of Computer Science, Drexel University, Philadelphia, PA 19104 – lgreenwald@cs.drexel.edu

KEY WORDS: Dynamic Planning, Automation, Real-Time Planning, Remote Sensing, Artificial_Intelligence

ABSTRACT:

Imaging satellites are traditionally scheduled in a static fashion; namely the schedule is created off-line and then uploaded to one or more imaging satellites to be executed as an immutable sequence of commands. In this paper, we make the case for dynamic scheduling of imaging satellites. Dynamic schedules will allow satellite systems to take advantage of information gathered during the execution of the schedule and react to changes in the environment, desired tasking, and the availability of resources. We develop the remote sensing scheduling problem and discuss contingency conditions under which the satellite scheduling problem becomes dynamic. We then review existing work on contingency scheduling and conditional scheduling and propose extensions to address the dynamic satellite scheduling problem. Dynamic schedules will yield improved mission schedules and reduced mission costs.

1. INTRODUCTION

Satellite scheduling is the process of assigning resources (Earth observing satellites, ground station antennae, *etc*.) to tasks (*e.g*., observations, uplinks, downlinks, control maneuvers, *etc*.). Satellites and satellite ground stations are expensive resources that are and will continue to be in high demand. Consequently, it is important to develop schedules that effectively maximize the return on investment. Current satellite scheduling practices fail to meet this objective in part because observation collection schedules are statically generated and then executed. In this paper, we focus on the role of explicitly modeling and reasoning about dynamics in the generation and execution of payload operations for remote sensing satellites.

The mission of remote sensing satellites is to collect data and then transmit the data to Earth. Commercial remote sensing satellites (*e.g*., SPOT, Ikonos, RadarSat, *etc*.) are in low Earth orbits, and consequently the ground area visible to the satellite changes over time. This movement creates visibility windows that affect the availability of satellites to perform tasks, including observation and communication tasks. Additional scheduling difficulties include the availability of resources to support planned maintenance and control maneuvers, as well as constraints on the timing and ordering of task execution.

Today, Earth observing satellite systems rely on ground systems software and ground station operators to pre-calculate a fixed set of observation tasks that are uploaded to the satellite as an immutable sequence of commands, oftentimes several hours prior to execution. Even if the upload event occurs closer to the actual execution, tasking is often frozen for several hours before the upload event. This characteristic of satellite operations is due to many factors, including the availability of opportunities to upload commands to the satellite, as well as the time consuming but important process of validating that scheduled tasks do not violate health and safety constraints of the satellite.

A problem with this method of satellite scheduling is that these immutable sequences of commands are brittle in that they have no mechanisms to gracefully recover when execution conditions vary from those anticipated during initial schedule creation. For example, if a scheduled downlink event is missed due to communication interference, either stored observation data is lost or subsequently scheduled observations cannot be completed successfully. In general, real-world variations to scheduling problems can take the form of changes in resource availability, last minute tasking, and feedback during the execution of a schedule. Static schedules, in the form of immutable sequences of commands, cannot respond quickly to either targets of opportunity (*e.g*., an unexpected volcanic eruption) or unexpected difficulties (*e.g*., clouds blocking a critical target).

In this paper, we propose to augment traditional open loop, fixed execution scheduling of remote sensing satellites with a contingency-based adaptive scheduling approach. This new technology requires a combination of 1) richer models of dynamics and uncertainty in satellite schedule execution, 2) limited on-satellite processing of schedule adaptations, and 3) real-time feedback on schedule execution and environmental conditions. The resulting dynamic scheduling solution will yield improved missions and reduced mission costs.

This paper is organized as follows. The next section contains an overview of the problem of scheduling remote sensing satellites. In the section on contingencies, we describe the conditions under which the remote sensing scheduling problem requires more dynamic solutions. In the section on contingency scheduling, we discuss several approaches to dynamic scheduling, including foundational work on the similar problems of telescope scheduling, Mars rover planning, and real-time avionics scheduling. In the final section we discuss the application of dynamic scheduling to satellite scheduling and present both ongoing and future work.

2. GENERAL SATELLITE SCHEDULING PROBLEM

A satellite scheduling problem can be modeled using four basic objects: tasks, resources, events and constraints. *Tasks* are the activities and operations to be performed. *Resources* are the people, satellites, sensors, *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 satellite schedule is a sequence of tasks $\boldsymbol{A}$, such that each task $A_i \in \boldsymbol{A}$ is assigned a set of one or more resources $R_i \in \boldsymbol{R}$, a start time, t_i, and duration, d_i. For example, a downlink task may be assigned a satellite antenna resource and a ground antenna resource, a start time and duration. A scheduling system is responsible for determining a set of assignments of resources to tasks, subject to the restrictions specified by events and constraints. In cases where the specific set of resources needed to perform a task are known *a priori*, the scheduling problem is reduced to finding a start time and duration for each combination of task and resource. When the tasks have a fixed duration, then the problem is further reduced to finding start times.

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 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 must also be scheduled to renew the resource (*e.g.*, down-linking data to the ground station frees up the satellite's memory).

Event constraints are used to describe time windows during which a task can be executed. One important difference between satellite scheduling problems and traditional scheduling problems is that all tasks (*e.g.* sensing, communications) for which low-Earth orbiting satellite resources must interact with ground locations must have associated event time windows. Satellites must be able to see a ground location in order to perform a given task. This feature introduces time-varying parameters to the scheduling problem; significantly complicating the scheduling problem through the need to model changes to resources that cannot be controlled by the scheduler itself.

Additional complexities of the general satellite scheduling problem include task setup times that are dependent on the ordering of tasks; periodic tasks such as maintenance checks that may be specified in terms of clock time or with a specification relative to other tasks; a mixture of pre-emptible and non-pre-emptible tasks and subtasks (*i.e.*, tasks with a non-pre-emptible initial activity followed by a pre-emptible period of activity); a mixture of renewable (*e.g.*, memory, battery power) and nonrenewable resources (*e.g.*, fuel); and specifications for schedule optimization criteria that vary from finding feasible schedules that do not violate constraints to finding schedules that observe task priority orderings. We present more detail on the general satellite scheduling problem and a search-based solution in [Pemberton, 2001].

A satellite scheduling system typically implements the following procedure sketch:

1. Construct and validate an initial schedule of activities
2. Allow the satellite operator to change the schedule, problem inputs (resources, resource availability, collection requests, *etc*.), and optimization criteria
3. Validate the new schedule or re-construct a schedule given new information
4. Upload the resulting static schedule to each satellite
5. Execute the static schedule until successful completion or failure
6. Incorporate feedback from prior execution and repeat

A weakness of systems implementing this sketch is in the inefficiencies introduced in Step 5. Any schedule failures detected in this step cannot be remedied until the entire process repeats. Furthermore, there are no mechanisms to exploit advantageous conditions unanticipated during schedule construction, or to permit re-tasking during execution. At best, this can introduce significant resource downtime, resulting in increased mission costs. At worst, the health of the satellites themselves could be at risk.

3. CONTINGENCIES

Static schedules, in the form of immutable sequences of commands, cannot respond quickly to either targets of opportunity (*e.g.*, an unexpected volcanic eruption) or unexpected difficulties (*e.g.*, clouds blocking a critical target). In general, we'd like to be able to construct schedules that automatically and rapidly respond to the following types of contingencies:

1. Targets of opportunity
2. Unexpected changes in the availability of resources
3. New (short notice) tasking

Furthermore, we'd like to explicitly model problem uncertainties and provide solutions that are flexible to execution-time variations in problem parameters.

3.1 Targets Of Opportunity

The best way to describe targets of opportunity is by an example. Consider an imaging satellite with two sensors: one low-resolution (wide area) sensor and one high-resolution (narrow area) sensor. The low-resolution sensor is used for initial detection of interesting events, while the high-resolution sensor permits detailed analyses of these events. For example, consider search and rescue at sea or after a plane crash. It is inefficient to schedule high-resolution imaging for all locations of potential interest. Thus a scheduling algorithm needs to schedule some combination of low-resolution and high-resolution imaging tasks. A static schedule may only make use of information available at schedule construction time in order to determine when to take low-resolution images and when to take high-resolution images. All images are down-linked and processed before new information can be used in schedule construction. The delay introduced during this process might lead to lost opportunities.

Now consider an imaging satellite with the ability to process low-resolution images on-board. A schedule may now consist of a combination of low-resolution imaging tasks, on-board

image processing tasks, and high-resolution imaging tasks. In order to make use of this new capability, we need a schedule that employs the output of on-board image processing to make decisions about whether or not to take high-resolution images. This cannot be done with a static schedule. The schedule must be flexible enough to permit automatic changes during execution. At the very least, the decision of whether to store and download an image can be made based on an initial image analysis. This will save both on-board memory storage and downlink capacity and possibly free up these resources to allow additional imaging before the next downlink opportunity.

Note that on-board processing is not strictly necessary in this scenario. It is also possible to achieve similar results with a combination of tightly coupled low-resolution imaging, downlinking, processing, uplinking, and feedback-driven high-resolution imaging. In both cases the scheduling system requires the ability to rapidly adapt the schedule based on execution-time feedback. Other target of opportunity examples include forest fire monitoring in which initial on-board processing of an infra-red sensor could determine the best targets for high-resolution imagery to better analyze the terrain and conditions for fire control; and the use of wide angle, low-resolution imaging to detect possible indications of a terrorist training facility that would trigger the need for additional high-resolution imagery to confirm the initial assessment.

3.2 Resource Changes

During the execution of a schedule, it is possible that some of the assigned resources become unexpectedly unavailable. Similarly, it is also possible that additional resources become unexpectedly available during schedule execution. For example, satellites, like all resources, also have unplanned outages such as those due to solar flares. Sometimes the effect of solar flares can be predicted and sometimes they can't. While a static schedule can take into account planned outages and deterministically predictable resource changes, non-deterministic resource changes such as those due to solar flares require a more flexible scheduling approach. A flexible schedule must adapt to both positive and negative changes to resource availability.

Consider an imaging sensor that becomes unavailable in conditions of thick cloud cover or a communications resource that is susceptible to radio interference. Weather and radio interference are difficult to predict deterministically at schedule construction time. However, they are easy to detect during schedule execution. A flexible schedule would make use of this feedback to dynamically alter the assignment of resources and start times to tasks. For example, on-board image analysis could detect the degree of cloud cover and modify the remaining schedule accordingly (*e.g.*, try the task again on this orbit pass if it is an isolated cloud or cancel other tasks on this part of the orbit if the cloud cover is extensive since other tasks in that area will likely fail a maximum cloud cover constraint).

3.3 New Tasking

Targets of opportunity and resource changes introduce the need to use execution-time feedback to adapt a satellite schedule. A different form of execution-time feedback is the ability to rapidly accommodate new, high priority tasks without going through a completely new cycle of scheduling and uplinking. The ability to accommodate customer feedback in this way greatly improves the effective use of a set of satellite resources. In order to satisfy the new task, other lower priority tasks will need to be removed from the current schedule. A flexible schedule would anticipate such potential schedule changes prior to initial uplink and provide opportunities for customer feedback that do not necessitate complete re-scheduling. For example, the insertion of a high priority task might cause previously scheduled but lower priority tasks to be moved into secondary schedule slots that still satisfy all constraints. The existence of such slots can be determined when the schedule is initially built in order to permit rapid introduction of new tasks.

3.4 Problem Parameter Uncertainty

We have already discussed how environmental uncertainty can affect targets of opportunity, resource changes, and new tasks. Additionally, there may be uncertainty in specification of the scheduling problem parameters. For example, the duration requirement of a communications task might be modeled based on the number of bytes to be transferred, distance between satellite and ground terminal, and communication properties. However, interference from other sources may lead to lost packets and subsequent re-transmissions by the communication protocol. The number of re-transmissions and thus, the communication duration, can only be modeled by a stochastic distribution. If this distribution is wide, a worst case specification would lead to assigning resources for longer durations than normally required. This leads to costly idle satellite resources. A flexible scheduling solution could detect when a task has successfully completed and modify remaining tasks in the schedule accordingly; executing tasks earlier to exploit shorter-than-expected durations and moving tasks later to accommodate longer-than-expected durations. Similarly, low priority tasks could be interrupted if extending their durations would make it impossible for higher priority tasks to be successfully executed.

Other problem parameters with execution-time variations include event time windows. Orbit propagation models convert position and velocity to time-ordered positions; sensor and antennae models combine with position to give visibility. While the orbital motion of a satellite can be predicted with high levels of precision, the ground visibility depends on highly varying sensor and antennae execution-time conditions. Ground visibility variations affect both sensing and communications. Communication variations are more complicated if we permit multi-hop communication paths using constellations of satellites. Power-sensitive tasks and set-up times have potentially large variations as well.

4. CONTINGENCY SCHEDULING

As discussed in the previous section, static satellite scheduling methods do not admit methods for handling contingencies such as targets of opportunity, unexpected changes in the availability of resources, or new short notice tasking. Furthermore, traditional static scheduling approaches do not explicitly model problem uncertainties nor provide solutions that are flexible to execution-time variations in problem parameters. This is consistent with the bulk of the literature on real-time scheduling, in which static schedules are designed to handle periodic processes. In this section we present recent scheduling methods that address the limitations of static scheduling when allocating resources in dynamic environments. Especially important are several approaches explicitly designed to accommodate dynamic information about the scheduling problem. In these approaches dynamic information is used as a

fundamental tool in determining how to alter schedules over time. These approaches have been designed to solve scheduling problems for telescopes, Mars rovers, and real-time avionics. In the next section we discuss the applicability of these methods to satellite scheduling and discuss our ongoing work.

4.1 Telescope Scheduling

In work on *just-in-case* scheduling [Drummond, 1994], a stochastic model of action duration is used to generate contingent schedules that address likely breaks in a nominal static schedule. For example, in the domain of telescope scheduling, a nominal schedule of telescope observations is constructed such that each observation takes place during a legal observation window, corresponding to the necessary conditions for correct observations. The amount of time an observation requires depends on the time it takes to position the telescope, center on the feature of interest, and take a reading. This time is a non-deterministic function of the night conditions, such as how clear the sky is. A schedule breaks if the amount of time required to take an observation exceeds the amount of time allocated by the nominal schedule (causing the subsequent observation to fail) or if the observation time is much less than that allocated (causing a wasteful and costly idling of telescope resources).

To proactively handle schedule breaks, just-in-case scheduling analyzes the nominal schedule off-line for potential breaks and creates branching points such that an alternative schedule is available if the nominal schedule breaks. The algorithm finds the most probable break point (assuming a uniform distribution over the range of possible task execution times), creates an alternative schedule starting from that break point, and repeats this process while time and space permit. The resulting contingent schedule improves the original static schedule by employing execution-time feedback about task completion to dynamically alter the schedule. The altered schedule makes use of expensive telescope time that would have gone idle as a result of the schedule break. In cases in which this method successfully anticipates schedule breaks, it avoids the computational costs of on-demand reasoning during time-critical on-line execution periods. Just-in-case scheduling has been observed to work well if the probability of successful execution of the nominal schedule leaves room for improvement (< 1.0), and there are a small number of high-probability schedule breaks, as opposed to a large number of evenly distributed breaks [Drummond, 1994].

Just-in-case scheduling is an extension of earlier work on *anytime synthetic projection* [Drummond, 1990]. This earlier work requires a discrete model for action outcome in incrementally constructing conditional plans for stochastic domains. Just-in-case scheduling permits action uncertainty to be modeled with continuous time durations. While just-in-case scheduling focuses on optimally scheduling resources under uncertainty,,synthetic projection is more concerned with maximizing the probability of goal achievement in planning.

4.2 Mars Rover Scheduling

The Mars rover autonomy architecture consists of multiple interacting components, distributed between Earth-based and Mars-based systems [Washington, 1999]. Planning and scheduling is performed on Earth-based processors with input from rover operators and scientists. Just-in-Case [Drummond, 1994] contingency plans and flexible schedules are up-linked to the on-board rover executive. The rover executive then communicates commands to the rover real-time control system while monitoring the environment and resource states to manage conflicts and take advantage of opportunities.

To increase autonomy, processing is being shifted from interactive Earth-based systems to rover on-board systems and Mars-based peripheral support systems. Researchers at NASA Ames [Bresina, 1999, Washington, 1999] have made a number of extensions to the just-in-case scheduling method in order to increase rover autonomy and to improve rover robustness, flexibility, resource utilization, and failure recovery. As an example of rover flexibility and autonomy, consider a rover traversing a ridge under the control of a time-stamped set of motion commands [Washington, 1999]. If, at some point, the rover's on-board navigation system fails to locate a safe path in the direction of the next motion command, the pre-conditions of that motion command fails causing the rover to invoke a contingent plan. Without contingency plans the rover would be stuck until the next down-link/up-link cycle; exposing the rover to risky environmental conditions. Contingency plans have their limits, however. At some point it becomes infeasible to construct and verify a large branching contingency plan off-line, as well as impractical to up-link the plan [Washington, 1999]. At this point on-line planning and plan adaptation capability becomes crucial, including the ability to delete plan steps and merge the current plan with alternate plans from an on-board plan library [Bresina, 2001]. However, on-board processing must still be conservative to avoid disastrous results.

Extensions to just-in-case scheduling include the ability to reason about schedule breaks caused by resources other than time, including power consumption. Additional improvements include modeling uncertainty with non-uniform distributions, modeling and reasoning about concurrent actions, and the development of improved ground tools [Washington, 1999]. Other related work includes work at NASA and JPL [Pell, 1997] on issues of safe execution during on-board planning.

4.3 Real-Time Avionics Scheduling

A crucial component of satellite scheduling is the ability to maintain safe and robust operation of satellite resources at all times. Solutions must ensure verifiably safe behavior. *Conditional scheduling* [Greenwald, 1998] makes use of a predictive model of the dynamic environment to replace on-board processing with the conditional sequencing of provably safe task-execution schedules constructed off-line. Contrasted with on-line planning, this approach permits tight off-line time and space guarantees.

Conditional scheduling was developed in the context of real-time avionics scheduling but has its origin in a general method for designing real-time control systems based on dynamically sequencing condition-specific controllers [Greenwald, 1997a and 1997b] as well as earlier work on the interaction of reaction and deliberation [Greenwald, 1994]. A real-time avionics schedule allocates the resources of a distributed multiprocessor system (*e.g.* [Carpenter, 1994, Hoyme, 1993]) to the execution of aircraft flight operations, such as thrust management, flight-data acquisition, flight management, and climate control. A schedule samples (executes) each flight operation at a specified frequency (within worst-case latency and jitter bounds). Providing provably safe schedules is complicated by uncertain in-flight conditions and system resources that are tightly limited by power and weight considerations.

The traditional approach to real-time avionics scheduling is to design a single static schedule that samples flight operations for fixed durations and at fixed frequencies throughout the flight. For static scheduling solutions, behavior and timing guarantees are dictated by worst-case in-flight conditions. Conditional scheduling replaces the static schedule with a collection of situation-specific schedules. Each schedule is designed to provide guaranteed behavior (*e.g.*, safety) over a subset of in-flight conditions. An execution component is provided to dynamically switch schedules as conditions change.

Conditional schedules permit more effective use of limited resources by taking advantage of situation-specific conditional relationships between flight operations. For example, the thrust-management operation may require frequent execution during take-off and landing, while stabilization may require more frequent execution during cruising. Additionally, climate control may need more monitoring during warm clear weather conditions, while the navigation control may require more frequent monitoring during foggy conditions. The flight conditions that dictate worst-case sampling frequencies may differ for different operations. Simultaneously sampling all flight operations at worst-case frequency may never be necessary in practice. Off-line analyses, including reachability analysis for predicting dynamically changing situations, are used to determine the required set of conditional schedules. Provably safe conditional schedules are then pre-compiled for specific in-flight conditions. The execution component must implement a strategy for switching schedules that guarantees that, at any given time, the active schedule controlling the avionics hardware implements sampling frequencies that are appropriate for any in-flight condition that may occur before a new schedule is activated. All computational resources required to execute this strategy must be explicitly modeled in order to provide real-time behavior guarantees.

While just-in-case scheduling builds a schedule and then analyzes the schedule for potential breaks, conditional scheduling anticipates errors by modeling the dynamic conditions that predict errors. The analytical techniques of conditional scheduling may be used to determine the level of achievable guaranteed behavior. Furthermore, conditional scheduling techniques may be used to reason about the bounded time and space available for executing a just-in-case scheduling solution. Conditional scheduling may be extended to proactively manage both variable sampling frequencies and variable action durations.

4.4 Other Approaches To Dynamic Scheduling

Dynamically altering schedules has been discussed in work on *mode changes* [Fohler, 1992, Sha, 1989] and *parametric dispatching* [Gerber, 1995]. A mode change consists of switching static schedules at execution-time in response to a specific event that triggers the change. A mode is analogous to a set of situations for which a static schedule may be designed in conditional scheduling. A mode change event corresponds to a transition from one set of situations to another. Modes and mode change events are manually specified as part of the off-line design problem. The focus is on safe operation of the system during the transition from one mode to the next. This work is complementary to conditional scheduling in that conditional scheduling focuses on providing off-line methods to help conceive, generate, and validate switching strategies that guarantee that any time delays caused by switching do not jeopardize execution-time behavior guarantees.

In parametric dispatching, tasks are statically ordered off-line but the actual start time and duration of each task are determined dynamically at execution time. Parametric dispatching consists of an off-line component to determine the static ordering, an execution-time scheduler to dynamically determine upper and lower bounds on start times for each task, and a fast dispatcher to implement the schedule, taking into account any non-real-time tasks. This method relies on execution-time scheduling to handle dynamic events, as opposed to a predictive model of dynamics employed in conditional scheduling. Parametric dispatching provides guarantees with respect to the static, deterministically known components of the schedule. Given nondeterministic task durations, a parametric dispatching solution cannot guarantee off-line any improvement over the worst-case static solution, though the expectation is that it will adapt to varying task durations. Given a predictive dynamic model, the analytical techniques developed for conditional scheduling may prove suitable for analyzing the execution-time components of parametric dispatching solutions as well.

5. DISCUSSION AND FUTURE WORK

Much of the recent work on contingency scheduling detailed in the previous section contains solutions or partial solutions to the dynamic satellite scheduling problem. We are currently in the process of evaluating the applicability of these methods.

Satellite scheduling problems involve multiple resources. As such, they require the extensions to just-in-case scheduling developed for the Mars rover rather than the single-resource solutions developed for the telescope scheduling problem. At the same time, the pre-supposition of on-board processing permitted for Mars rovers may not be cost effective in fuel-constrained satellites. The conditional scheduling approach to minimizing on-board processing to conserve power and weight may be more applicable. Furthermore, the conditional scheduling focus on off-line validation is also important for satellite scheduling. Conditional scheduling is well-suited for problems in which potential new tasks or potential resource outages may be anticipated long before the start of the execution of the schedule, .Contingency scheduling or a more adaptive scheduling approach may be better suited for problems in which a new unanticipated task appears or resource fails during the execution of the schedule. Thus, some combination of mechanisms developed for Mars rover scheduling and real-time avionics seems most promising.

New capabilities are required in order to augment traditional open loop, fixed execution scheduling of remote sensing satellites with contingency-based adaptive scheduling. These include 1) richer models of dynamics and uncertainty in satellite schedule execution, 2) limited on-satellite processing of schedule adaptations, and 3) real-time feedback on schedule execution and environmental conditions. In developing models of dynamics and uncertainty we can make use of existing models for orbit propagation, antennae, and sensors to derive communication and sensor visibility models

While many of the complexities of satellite scheduling have been addressed in recent work on contingency scheduling, there are a few important problem features that introduce new complexities. One new complexity is providing the ability to differentiate priority levels for satellite tasks using an ordinal scale. For example, breaks in high priority tasks must be treated differentially from breaks in low-priority tasks, even if

the break probability for the latter is higher. Devising a numeric scale for these ordinal priorities would permit decision-theoretic trade-offs. A second new complexity is removing the assumption of independent schedule breaks. It may be necessary to model problems for which there is a part of the schedule where several high priority tasks will get bumped if a new task is added or a vital resource fails. Finally, the periodic dynamics of orbit propagation models is a new problem feature that we plan to exploit in developing new contingency scheduling methods. For example, these predictable dynamics can inform switching strategies in solutions based on conditional scheduling.

The communication issues of Mars rover operation moves that work in the direction of on-board autonomy. With satellite scheduling it is possible to conceive a solution in which scheduling functions are shared between ground-based and space-based processing. In this case issues of time lag in closing the communications loop need to be addressed

While future satellites will have additional capability for intelligent behavior, the costs of on-board processing will limit the level of autonomy in the near future. We anticipate incremental improvements in on-board autonomy and the ability to coordinate processing across teams of satellites with reduced levels of ground-based support. The first step will be the ability of the satellite to recognize crude features of the raw collection data. For example, sensors may be able to detect cloud cover that wasn't previously predicted and reassign the imaging task to a later opportunity. Eventually, satellites will be able to pass an imaging task to other satellites that could observe a missed target. Providing satellites with the ability to process collection data on-board will allow them to dynamically prioritize collection tasks to achieve a higher-level goal. For example, a satellite with an infrared sensor and a high-resolution imaging sensor might scan a target area with the infrared sensor to identify valuable targets for collection on the same pass. We are exploring this approach in the context of teams of satellites coordinating observation activities with each other as well as with ground-based stations. In this scenario we must provide observation and communication policies for each satellite. Each satellite has local knowledge and requires an observation policy that dynamically targets interesting areas. Overlapping observation fields (in time and space) lead to coordination issues, and subsequent communication and synchronization issues. Policies must trade-off the cost of communication with the cost of redundant observations or missed observation opportunities. Communication may be among satellite teams or between ground-stations and satellites. Maximum autonomy is achieved when satellites construct and implement their own observation and communication policies. Minimal autonomy requires a ground-based system to coordinate all observations.

We are in the process of further specifying the dynamic satellite scheduling problem in the context of the described work on contingency scheduling. Future work includes experimenting with contingency scheduling solutions. Simulation tools exist for aiding these studies. We anticipate that this rich area of research will result in dynamic scheduling solutions that yield improved missions and reduced mission costs.

REFERENCES

Bresina, J., Golden, K., Smith, D.E., Washington, R., 1999. Increased flexibility and robustness for Mars rovers. In *Fifth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, ESTEC, Noordwijk, Netherlands.

Bresina, J., Washington, R., 2001 Robustness via Run-time Adaptation of Contingent Plans. In *Proceedings of the AAAI-2001 Spring Symposium: Robust Autonomy*, Stanford, CA.

Carpenter, T., Driscoll, K., Hoyme, K., Carciofini, J., 1994 ARINC 659 Scheduling: Problem Definition. *Proceedings of the 15th IEEE Real-Time Systems Symp.*, San Juan, Puerto Rico.

Drummond, M., Bresina, J., 1990 Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. *Proceedings AAAI-90*, pp. 138-144, Boston, MA.

Drummond, M., Bresina, J., Swanson, K., 1994. Just-in-case scheduling. In *Proceedings AAAI-94*.

Fohler, G., 1992. Realizing changes of operational modes with a pre run-time scheduled hard real-time system. In *Second International Workshop on Responsive Computer Systems*.

Gerber, R., Pugh, W., Saksena, M., 1995. Parametric dispatching of hard real-time tasks. *IEEE Transactions on Computers*. 44(3).

Greenwald, L., 1997a. *Analysis and Design of On-line Decision-Making Solutions for Time-Critical Planning and Scheduling Under Uncertainty*. Ph.D. Dissertation, Brown University, Providence, RI.

Greenwald, L., Dean, T., 1994. Solving time-critical decision-making problems with predictable computational demands. In *Second International Conference on AI Planning Systems*.

Greenwald, L., Dean, T., 1997b. Tradeoffs in the design of on-line systems. In *Working Notes of AAAI-97 Workshop on On-Line Search*, Providence, RI.

Greenwald, L., Dean, T., 1998. A conditional scheduling approach to designing real-time systems. In *Fourth International Conference on AI Planning Systems*.

Hoyme, K., Driscoll, K., 1993. SAFEbus(tm). *IEEE Aerospace Electronics and Systems Magazine*, 34-39.

Pell, B., Gat, E., Keesing, R., Muscettola, N., Smith, B., 1997. Robust periodic planning and execution for autonomous spacecraft. In *Proceedings IJCAI 15*, pages 1234—1239.

Pemberton, J., Galiber F., 2001. A Constraint-Based Approach to Satellite Scheduling. In E. Freuder and R. Wallace, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Constraint Programming and Large Scale Discrete Optimization* 57:101-114.

Sha, L., Rajkumar, R., Lehoczky, J., Ramamritham, K., 1989. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems* 1(3):243—265.

Washington, R., Golden, K., Bresina, J., Smith, D.E., Anderson, C., Smith, T., 1999. Autonomous rovers for Mars exploration. In *Proceedings of the 1999 IEEE Aerospace Conf.*, Aspen, CO.

ACKNOWLEDGMENTS

This work has been supported in part by the Air Force and ARPA under grant No. F30602-95-1-0020, by NSF in conjunction with ARPA under grant No. IRI-9312395, and by the Veterans Health Administration through a post-doctoral fellowship. This work has also been supported in part by Veridian Internal Research and Development funding.