Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft

Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
{firstname.lastname}@jpl.nasa.gov

Abstract

An autonomous spacecraft must balance long-term and short-term considerations. It must perform purposeful activities that ensure long-term science and engineering goals are achieved and ensure that it maintains positive resource margins. This requires planning in advance to avoid a series of shortsighted decisions that can lead to failure. However, it must also respond in a timely fashion to a somewhat dynamic and unpredictable environment. Thus, spacecraft plans must often be modified due to fortuitous events such as early completion of observations and setbacks such as failure to acquire a guidestar for a science observation. This paper describes the use of iterative repair to support continuous modification and updating of a current working plan in light of changing operating context.

Introduction

In recent years Galileo, Clementine, Mars Pathfinder, Lunar Prospector, and Cassini have all demonstrated a new range of robotic missions to explore our solar system. However, complex missions still require large teams of highly knowledgeable personnel working around the clock to generate and validate spacecraft command sequences. Increasing knowledge of our Earth, our planetary system, and our universe challenges NASA to fly large numbers of ambitious missions, while fiscal realities require doing so with budgets far smaller than in the past. In this climate, the automation of spacecraft commanding becomes an endeavor of crucial importance.

This paper describes an advance in automated planning and scheduling technology to spacecraft mission operations. This technology is applicable to a large spectrum of missions, from those that have very limited on-board computational capabilities (such as Lunar Prospector) to those that fly highly sophisticated software (such as Cassini). In all cases the goal is for the project science team to be able to command the spacecraft directly with no mission operations specialists involved in routine activities. In the most sophisticated missions the spacecraft operates autonomously, interacting with the ground systems and personnel only when it needs to schedule a downlink activity to transmit science data back to Earth. Autonomous spacecraft are made possible by equipping the spacecraft with sophisticated on-board software that provides knowledge and reasoning procedures to determine appropriate actions that achieve mission goals, to monitor spacecraft health during execution, and to recover autonomously from possible faults [9]. An on-board planner/scheduler is a key component of such a highly autonomous system. More generally, routine use of automated planning/scheduling systems for spacecraft operations, both in ground operations and on-board in an autonomous spacecraft, will have great impact on mission operations. Specifically, automated planning and scheduling provides the following benefits:

The extremely costly sequencing elements of the mission operations team would almost be eliminated, dramatically reducing cost. One estimate [10] indicated that automation of the commanding process could reduce mission operations costs by as much as 60% (excluding data analysis). Recent experiences support these projections. For example, use of the DATA-CHASER automated planning and scheduling system (DCAPS) to command the DATA-CHASER shuttle payload reduced commanding-related mission operations effort by 80% [3] as compared to manual generation of sequences.

Using planning and scheduling technology, a goal-based spacecraft could perform opportunistic science. When an unexpected opportunity occurs (such as a supernova or solar phenomena), the spacecraft could immediately respond by performing appropriate measurements rather than waiting until ground-based detection of the event and subsequent uplink of commands to the spacecraft.

A goal-based autonomous spacecraft could also enable interactive science, when appropriate. A self-commanding spacecraft could perform high-level science requests such as "Perform an interferometry sweep with priority 5." A direct connection between the scientist and spacecraft with faster feedback allows a new paradigm for scientific discovery in space.

Automated planning and scheduling technology offers the potential to increase science return by producing operations plans that better optimize use of scarce science resources.

For example, the DCAPS planner/scheduler increased science return by 40% over manually generated sequences [3]. This increase was mostly due to the short turn-around times (approximately 6 hours) imposed by operations constraints. This limited time did not allow for lengthy, manual optimization.

Finally, planning and scheduling technology simplifies the self-monitoring, onboard fault-management, and spacecraft health tasks. Because the spacecraft would be able to respond in a more goal-oriented fashion without the time lags introduced by ground communication, it is possible to cover a greater range of faults.

The remainder of this paper is organized as follows. First, we briefly describe the motivation for reducing planning response time in spacecraft operations. Next, we describe our technical approach to interleaving planning and execution to reduce this response time. We then follow with a description of our implemented architecture. Then we describe a mission scenario from the New Millennium Space Technology Four mission which was used to test our approach. Finally, we describe related work and ongoing efforts to further extend and validate this technology for future space missions.

Integrating Planning and Execution

An autonomous spacecraft must respond in a timely fashion to a (somewhat) dynamic, unpredictable environment. In terms of high-level, goal-oriented activity, spacecraft plans must often be modified in the event of fortuitous events such as observations completing early and setbacks such as failure to acquire a guidestar for a science observation. We call this situation *dynamic planning*, in which a plan must be continually updated in light of changing operating context. In such an operations mode, a planner would accept activity and state updates on a one to ten second time scale. Making the planner more timely in its responses has a number of benefits:

- The planner can be more responsive to unexpected (i.e., unmodelable) changes in the environment that would manifest themselves as updates on the execution status of activities as well as monitored state and resource values.
- The planner can reduce reliance on predictive models (e.g., inevitable modeling errors), since it will be updating its plans continually.
- Fault protection and execution layers need to worry about controlling the spacecraft over a shorter time horizon (as the planner will replan within a shorter time span).
- Because of the hierarchical reasoning taking place in the architecture there is no hard distinction between planning and execution – rather more deliberative (planner) functions reside in the longer-term reasoning horizons and the more reactive (execution) functions reside in the short-term reasoning horizons. Thus, there is no planner to executive translation process.

This introduction of the planner into the short-term planning horizon can also be motivated by current operations scenarios taken from the Space Infra-red Telescope Facility (SIRTF) [9]. In this operations scenario, the observatory is in a near-earth orbit and has a set of observation targets and their prioritizations. It is difficult to project exactly how future execution of the plan will proceed. For example, if a spacecraft is able to acquire the target quickly (as compared to conservative settling times and time to search for the target), an observation may complete significantly ahead of schedule. Alternatively, if the spacecraft repeatedly fails to acquire a guidestar required by an observation, the observation may be terminated. This also has the effect of completing the activity ahead of schedule but with a failed outcome. Within this operations context, a short-term planner would decide which observations to sequence next. Such a planner would need to consider all targets currently on the observation list, their visibility windows, and their relative positions in the sky (for reasons of slew minimization and for observation quality issues). The short-term planner would also need to track other resource management issues such as data management relating to engineering and science observations and coordination with downlink windows.

In a traditional plan-sense-act cycle, planning is considered

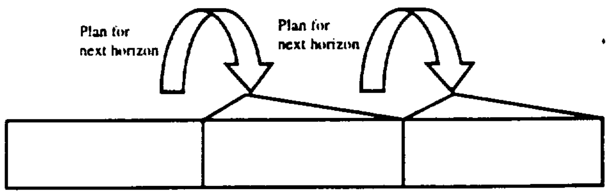


Figure 1 Traditional Batch Plan then Execute Cycle

a batch process and the system operates on a relatively long-term planning horizon. For example, operations for a spacecraft would be planned on the ground on a weekly or daily basis. In this mode of operations, the spacecraft state at the start of the planning horizon would be determined (typically predicted as the construction of the weekly plan would need to begin significantly before the week of execution). The science and engineering operations goals would then be considered, and a plan for achieving the goals would be generated. This plan or sequence would then be uplinked to the spacecraft for execution. The plan would then be executed onboard the spacecraft with little or no flexibility. If an unexpected event occurred due to environmental uncertainty or an unforeseen failure occurred, the spacecraft would be taken into a safe state by fault protection software. The spacecraft would wait in this state until the ground operations team could respond and determine a new plan. Correspondingly, if an unpredictable fortuitous event occurs, the plan cannot be modified to take advantage of the situation.

One model for operations is to move such planning and replanning functionality onboard, but to continue using it as a batch process. In this case, in the event of a fault, environmental event, or fortuitous event, the spacecraft can respond by entering into a stable state and replanning.

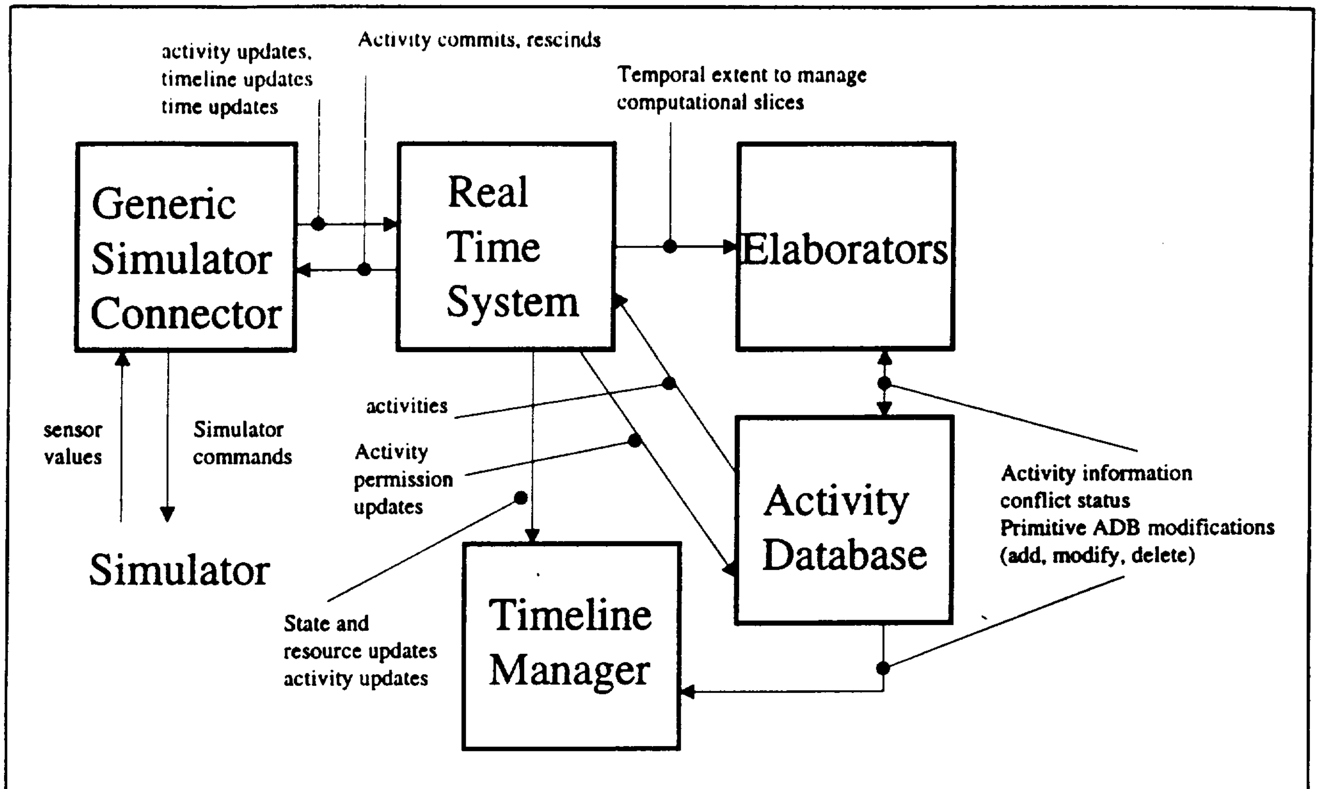


Figure 2 Overall Architecture for Continuous Planning

However, constructing a plan from scratch can be a computationally intensive process and onboard computational resources are typically quite limited, so that it still may require considerable time to generate a new operations plan. As a data point, the planner for the Remote Agent Experiment (RAX) flying on-board the New Millennium Deep Space One mission [11] is expected to take approximately 4 hours to produce a 3 day operations plan. RAX is running on a 25 MHz RAD 6000 flight processor and uses roughly 25% of the CPU processing power. While this is a significant improvement over waiting for ground intervention, making the planning process even more responsive (e.g., on a time scale of seconds) to changes in the operations context, would increase the overall time for which the spacecraft has a consistent plan. As long as a consistent plan exists, the spacecraft can keep busy working on the requested goals.

To achieve a higher level of responsiveness in a *dynamic planning* situation, we utilize a *continuous planning* approach and have implemented a system called CASPER (for Continuous Activity Scheduling Planning Execution and Replanning). Rather than considering planning a batch process in which a planner is presented with goals and an initial state, the planner has a current goal set, a plan, a current state, and a model of the expected future state. At any time an incremental update to the goals or current state may update the current state of the plan and thereby invoke the planner process. This update may be an unexpected event or simply time progressing forward. The planner is then responsible for maintaining a consistent, satisficing plan with the most current information. This current plan and projection is the planner's estimation as to what it expects to happen in the world if things go as expected. However, since things rarely go exactly as expected, the planner stands ready to continually modify the plan. Current iterative repair planning techniques enable incremental changes to the goals and the initial state or plan and then iteratively resolve any conflicts in the plan. After each update, its effects will be propagated through the current projections, conflicts identified, and the plan updated (e.g., plan repair algorithms invoked).

An Architecture for Integrated Planning and Execution

The overall architecture for the continuous planning approach is shown in Figure 2. The basic algorithm is as follows:

```
Initialize P to the null plan
Initialize G to the null goal set
Initialize S to the current state

Given a current plan P and a current goal set G

1. Update G to reflect new goals or goals that are
   no longer needed
2. Update S to the revised current state
3. Compute conflicts on (P,G,S)
4. Apply conflict resolution planning methods to
   P (within resource bounds)
5. release relevant near-term activities in P to
   RTS for execution
6. Goto 1
```

In this approach, the real-time software produces updates that require responses by near and long-term activities for the spacecraft. The spacecraft state is modeled by a set of timelines, which represents the current and expected evolution of the spacecraft over time. This model includes the current state (S) and the projection of how the state will evolve in light of actions expected to take place in the future. These actions are the current plan (P) that is also reflected in the timelines as actions at future points in time.

At each iteration through the loop shown above, as the world changes, the actual state of the spacecraft drifts from the state expected by the timelines. The real-time software updates the timeline models (S) with notifications of actual state values, actual resource values, actual start times, and completion times for activities. Each of these updates, when synchronized with the current plan may introduce conflicts (Step 3 above). A conflict occurs when an action in the plan is inappropriate – because its required state and/or resource values violate the system constraints.

Whenever such a conflict exists the planner notes the conflict and performs plan modifications to make the plan consistent with the current state and future projections. Because this process is continuous, the plan rarely has the opportunity to get significantly inconsistent. As a result the high-level actions of the system are more responsive to the actual spacecraft state. Also, planner activities at the lowest level directly correspond to commands to the simulator. The Generic Simulator Connector (Figure 2) handles the mapping from activities to simulator commands.

In conjunction with this incremental, continuous planner approach, we are also advocating a hierarchical approach to planning. In this approach, the long-term planning horizon is planned only at a very abstract level. Shorter and shorter planning horizons are planned in greater detail, until finally at the most specific level the planner plans only a short time in advance (just in time planning). This paradigm is shown in Figure 3.

The idea behind this hierarchical approach is that only very abstract projections can be made over the long-term and that detailed projections can only be made in the short-term because prediction is difficult due to limited computational resources and timely response requirements. Hence there is little utility in constructing a detailed plan far into the future – chances are it will end up being re-planned anyway. At one extreme the short-term plan may not be "planned" at all and may be a set of reactions to the current state in the context of the near-term plan. This approach is implemented in the control loop described above by making high-level goals active regardless of their temporal placement, but medium and low-level goals are only active if they occur in the near future. Likewise, conflicts are only regarded as important if they are high-level conflicts or if they occur in the near future. As the time of a conflict or goal approaches, it will eventually become active and the elaboration/planning process will then be applied to resolve the problem.

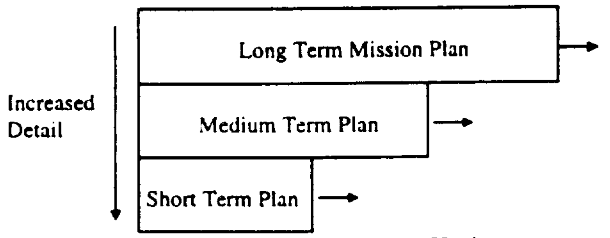


Figure 3 Hierarchical Planning Horizons

ST4 Spacecraft and Landed Operations Description

Deep Space 4 / Champollion (ST4) will be the fourth interplanetary spacecraft in NASA's New Millennium Program to identify, test, and fly advanced technologies onboard interplanetary spacecraft and Earth-orbiting satellites. In late 2005, following a two-and-a-half-year journey, ST4 will match orbits, or rendezvous, with Comet Tempel 1, as the comet is moving away from the Sun. The spacecraft will spend several months orbiting the comet nucleus, making highly accurate maps of its surface and making some preliminary compositional measurements of the gas in the coma. The data returned from ST4 will be used to determine the mass, shape, and density of the comet's nucleus and to make some early estimates about its composition.

After studying the nucleus from orbit, the spacecraft will send a small vehicle (a lander) to the surface. The touchdown itself will be quite tricky because scientists do not know whether the surface of the comet nucleus is hard, rocky, and rough, or soft and fluffy. Therefore, the challenge engineers face in designing the technology and instruments for this spacecraft is to be prepared for the unexpected. One of the ways ST4 engineers are preparing for all possible scenarios is by developing technologies to anchor the lander into the comet's surface no matter what its composition. Because the gravity of the comet nucleus is so weak, the lander must be anchored to the surface to permit drilling and sampling.

Once firmly in place, the lander will use a one-meter long drill to collect samples and then feed them to a gas chromatograph/mass spectrometer onboard the lander. This instrument will analyze the composition of the nucleus

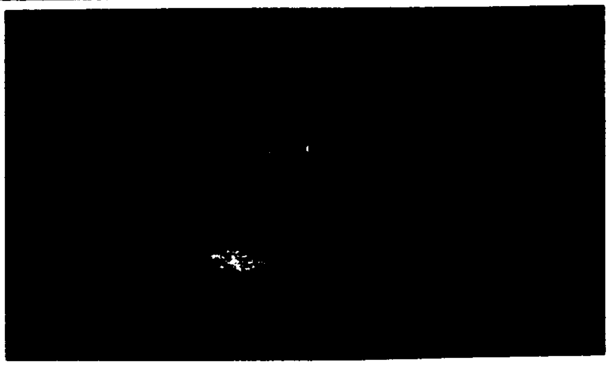


Figure 4 Artist depiction of ST4 lander landing on Comet

collected from various depths below the surface. The lander will also carry cameras to photograph the comet surface. Additional instruments planned onboard the lander to determine the chemical makeup of the cometary ices and dust will include an infrared/spectrometer microscope and a gamma-ray spectrometer. After several days on the surface, the lander will bring a sample back to the orbiter for return to Earth.

Continuous Planner ST4 Scenario

In order to test our integrated planning and execution approach, we have constructed a number of test cases within the ST4 landed operations scenario. We have also constructed a ST4 simulation, which accepts relatively high-level commands such as: MOVE-DRILL, START-DRILL, STOP-DRILL, TAKE-PICTURE, TURN-ON <device>, etc. The simulator also accepts scenario-time-control commands such as STEP, FFWD, and WARP. The simulation covers operations of hardware devices. In this test scenario the planner has models of 11 state and resource timelines, including drill location, battery power, data buffer, and camera state. The model also includes 19 activities such as uplink data, move drill, compress data, take picture, and perform oven experiment.

The continuous planner scenario has focused on the comet lander portion of the ST4 mission. It comprises a period of approximately 80 hours of lander operations on the comet surface. It is intended to represent a class of test cases against which to evaluate the performance of various command and control strategies for this portion of the mission.

The nominal mission scenario consists of three major classes of activities: drilling and material transport, instrument activity including imaging and in-situ materials experiments, and data uplink. Of these, drilling is the most complex and unpredictable.

The mission plan calls for three separate drilling activities. Each drilling activity drills a separate hole and acquires samples at three different depths during the process: a surface sample, a 20 cm. deep sample, and a one-meter deep sample. Acquiring a sample involves five separate "mining" operations after the hole has been drilled to the desired depth. Each mining operation removes 1 cm. of material. Drilling rate and power are unknown a priori, but there are reasonable worst-case estimates available. Drilling can fail altogether for a variety of reasons.

One of the three drilling operations is used to acquire material for sample-return. The other two are used to supply material to in-situ science experiments onboard the lander. These experiments involve depositing the samples in an oven, and taking data while the sample is heated. Between baking operations the oven must cool, but there are two ovens, allowing experiments to be interleaved unless one of the ovens fails.

We apply CASPER to this scenario to demonstrate three capabilities: 1) the ability to replan due to exogenous state conflicts (such as equipment failures), 2) the ability to replan due to exogenous resource conflicts (such as over-subscription of memory buffers), 3) and the ability to replan due to activity updates (such as drilling finishing late.)

One of the continuous planner capability to replan to perform a resource substitution after a component failure (Objective 1). The three planned sample activities each use oven 1 for baking the comet samples. During the simulation run, a failure was injected on oven 1. This changed the oven 1 state to "failed" for the remainder of the simulation. Because the second and third sample activities (as planned) use oven 1, these sample activities are in conflict because the sample activities require an operational oven (but are planned to use a "failed" oven). The planning system recognizes this conflict as a state required by an activity being different from the actual (or projected) state. The planner then attempts several fixes, including finding an activity to change the incorrect state. Unfortunately, there are no such activities to "fix" the oven. However, the sample activities require an oven resource, and there are two ovens on the ST4 lander. Hence the planner is able to find a repaired plan in which the second and third samples use oven 2 (see Figure 5.) The planning system could also have deleted the activity in conflict. However, the prioritization with the repair algorithm always considers moving or adding other activities to solve the conflict before deleting the conflicting activity.

Another continuous planner capability is to replan when a aggregate resource is over-subscribed or under-utilized (Objective 2). The data collected during the sample activities is compressed and then stored in the data buffer of the lander. This data is uplinked to the orbiting spacecraft at a later time. The planner uses estimates of the amount of data compression to plan when uplink activities are necessary. Because the compression algorithms are content dependent, these estimates may significantly deviate from actual achieved compression.

In this scenario, the actual data generated by the second

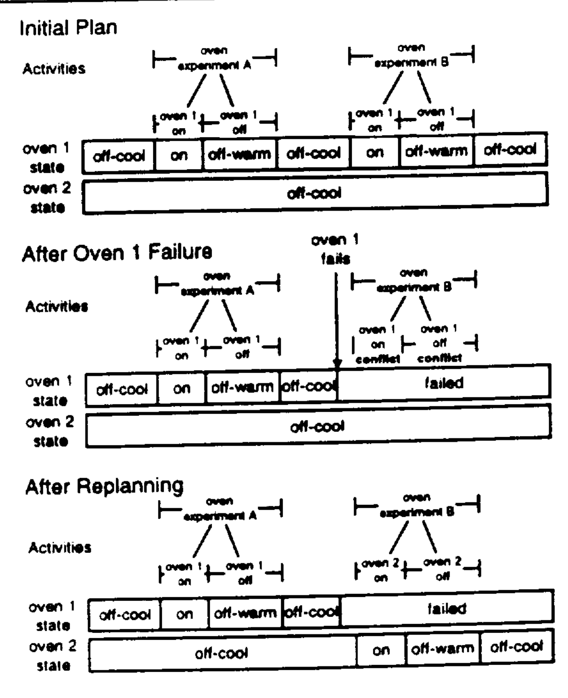


Figure 5 Oven State Example

sample activity is greater than expected because the compression achieved is less than originally estimated. The planner realizes that it will not have sufficient buffer memory to perform the third sample activity. This results in an over-subscription of the data buffer depletable resource. The planner knows that such a conflict can be repaired by: 1) removing activities that contribute to resource usage or 2) adding an activity which renews the resource. In this case these two options correspond to deleting the third sample activity or adding an uplink activity. (The uplink activity renews the buffer resource by uplinking data to the orbiter.) The planner resolves this conflict by adding an uplink activity after the second sample activity, freeing memory for the third sample activity (see Figure 6.)

Antoher demonstrated CASPER capability is to replan based on activity parameter updates (Objective 3). In the scenario, the mining operation using the drill takes longer than expected. This delays the oven experiment because no sample is yet prepared. The actual conflict is a violation of the temporal relationship between the mining activity and the oven experiment activity. (Mining must be completed before we continue to the oven experiment; see Figure 7.) In this example, the planner moves the oven experiment activity in order to repair this conflict.

Figure 8 contains a screen snapshot of the continuous planner prototype. The display is time oriented; later times are shown to the right on the horizontal axis. The upper portion of the screen shows the current activities in the mission plan, with each line beginning at the activity's start time and ending at its end time. The timelines toward the bottom of the display show the state and resource evolution as modeled and tracked by the planner.

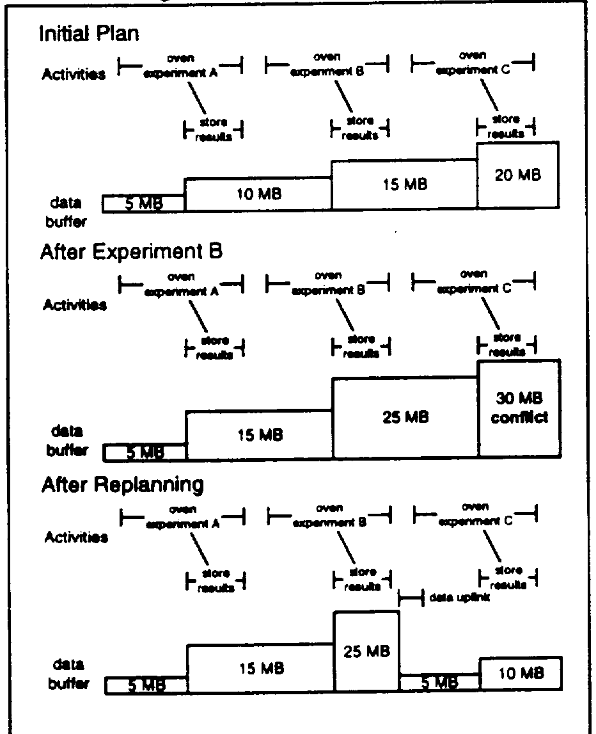


Figure 6 Over-subscribed Data Buffer Example

Discussion

While the current prototype has been tested on a range of cases in which state updates require replanning, all of the cases thus far have been ones in which the updates cause conflicts in the plan. In the case of the failed oven, buffer over-use, and activity completion time problem, the state update (when propagated through the plan) causes a conflict. There are other cases in which a state update enables a plan improvement. For example,

- battery power usage might be lower than expected enabling insertion of an additional sample activity content-dependent compression might perform better than expected allowing storage of additional experiment data; or
- drilling might be faster than expected again allowing for additional science activities.

In each of these cases, the planner needs to be aware of the potential for improvement in the current plan and be triggered to attempt to take advantage of the fortuitous situation. Our current prototype does not take advantage of these opportunities and we are slating this as future work.

In the current prototype, the planner can only respond to unexpected changes on activity boundaries. This can be limited in the context of activities with extremely long durations. This is because the planner does not have a model detailed enough to predict the resultant state if activities are interrupted in mid-execution. It would be useful if the planner could incorporate a model that could represent interruptible activities and act appropriately.

While we have tested our prototype on a range of scenarios, the test set has been quite small. We are currently working on enlarging the test suite and enhancing the simulation to address issues of noise in the simulation and commanding as well as approximate state estimation. These additional issues will further stress the architecture and are expected to lead to further insights and work.

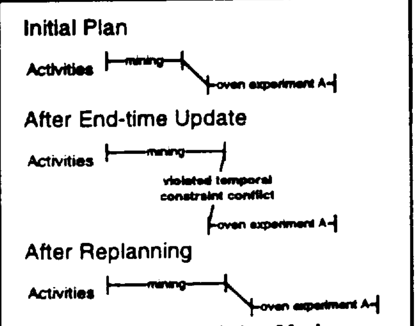
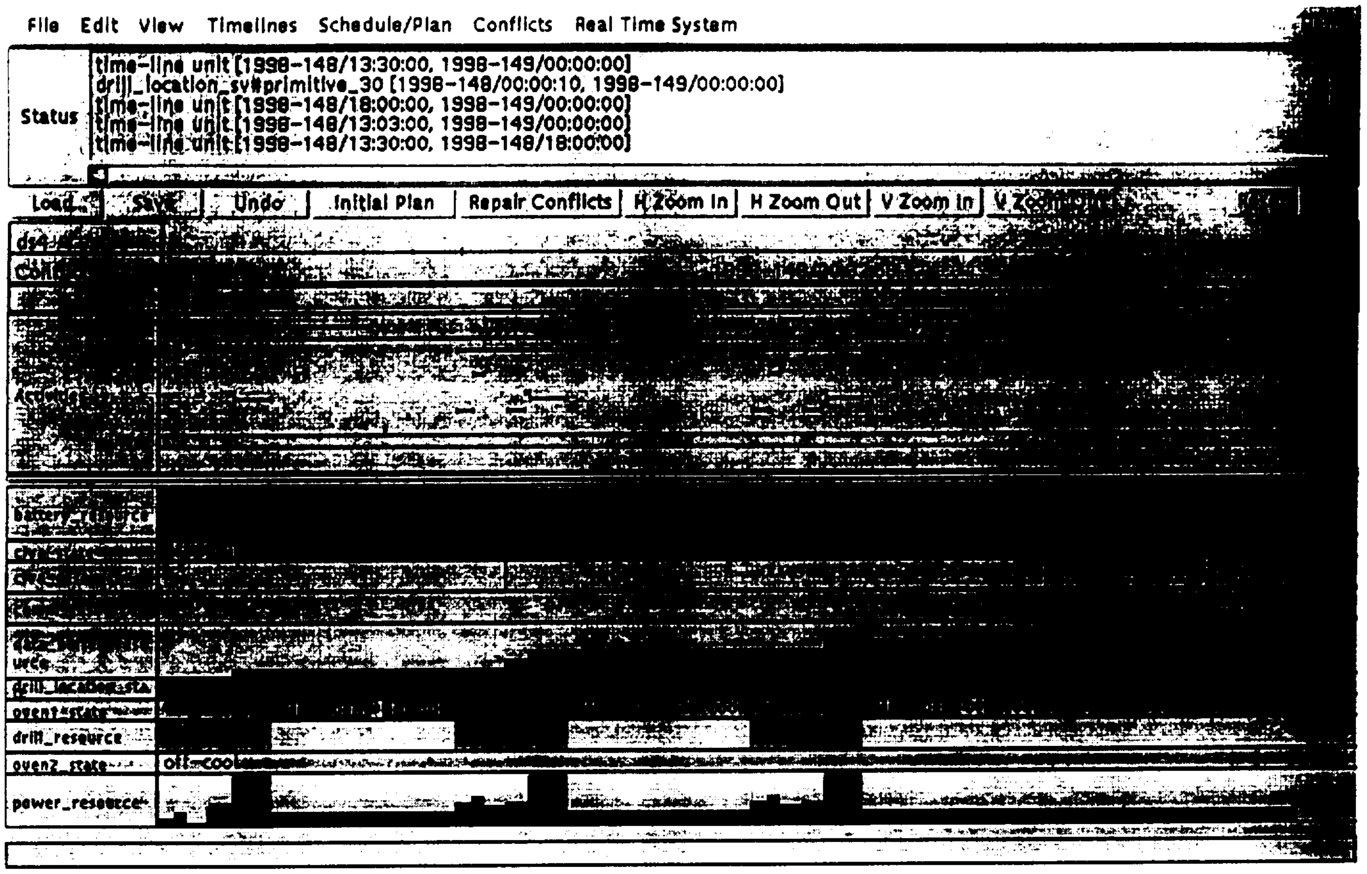


Figure 7 Activity Update Example



time-line unit [1998-148/13:30:00, 1998-149/00:00:00]
drill_location_sv#primitive_30 [1998-148/00:00:10, 1998-149/00:00:00]
time-line unit [1998-148/18:00:00, 1998-149/00:00:00]
time-line unit [1998-148/13:03:00, 1998-149/00:00:00]
time-line unit [1998-148/13:30:00, 1998-148/18:00:00]

Figure 8 Screen Snapshot Showing Oven Failed State before Replanning

This work builds on considerable previous work in iterative repair problem solving. The high-speed local search techniques used in our continuous planner prototype are an evolution of those developed for the DCAPS system [3] that has proven robust in actual applications. In terms of related work, iterative algorithms have been applied to a wide range of computer science problems such as traveling salesman [9] as well as Artificial Intelligence Planning [2, 6, 14, 16]. Iterative repair algorithms have also been used for a number of scheduling systems. The GERRY/GPSS system [17, 4] uses iterative repair with a global evaluation function and simulated annealing to schedule space shuttle ground processing activities. The Operations Mission Planner (OMP) [1] system used iterative repair in combination with a historical model of the scheduler actions (called chronologies) to avoid cycling and getting caught in local minima. Work by Johnston and Minton [7] shows how the min-conflicts heuristic can be used not only for scheduling but also for a wide range of constraint satisfaction problems. The OPIS system [15] can also be viewed as performing iterative repair. However, OPIS is more informed in the application of its repair methods in that it applies a set of analysis measures to classify the bottleneck before selecting a repair method. With iterative repair and local search techniques, we are exploring approaches complementary to backtracking refinement search approach used in the New Millennium Deep Space One Remote Agent Experiment Planner [11].

This paper has described an approach to integrating planning and execution for spacecraft control and operations. This approach has the benefit of reducing the amount of time required for an onboard planning process to respond to changes in the environment or goals. In our approach, environmental changes or inaccurate models cause updates to the current state model and future projections. Additionally, the planner's current goal set may change. In either case, if these changes matter (e.g., the current plan no longer applies) they will cause conflicts in the current plan. These conflicts are attacked using fast, local search and iterative repair methods

Acknowledgements

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Portions of this work were supported by the Autonomy Technology Program, managed by Dr. Richard Doyle and with Melvin Montemerlo as the headquarters program executive, NASA Code SM and by the Mission Data Systems Project, managed by Allan Sacks, NASA Code S. Comments and feedback from Bob Rasmussen, Kim Gostelow, Dan Dvorak, Erann Gat, Glenn Reeves, and Ed Gamble were very helpful in formulating and refining the ideas presented in this paper.

REFERENCES

[1] E. Biefeld and L. Cooper, "Bottleneck Identification Using Process Chronologies," *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[2] S. Chien and G. DeJong, "Constructing Simplified Plans via Truth Criteria Approximation," *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, June 1994, pp. 19-24.

[3] S. Chien, G. Rabideau, J. Willis, and T. Mann, "Automating Planning and Scheduling of Shuttle Payload Operations," *Artificial Intelligence Journal Special Issue on Applications*, 1998.

[4] M. Deale, M. Yvanovich, D. Schnitzius, D. Kautz, M. Carpenter, M. Zweben, G. Davis, and B. Daun, "The Space Shuttle Ground Processing System," in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.

[5] A. Fukunaga, G. Rabideau, S. Chien, D. Yan, "Towards an Application Framework for Automated Planning and Scheduling," *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space*, Tokyo, Japan, July 1997.

[6] K. Hammond, "Case-based Planning: Viewing Planning as a Memory Task," Academic Press, San Diego, 1989.

[7] M. Johnston and S. Minton, "Analyzing a Heuristic Strategy for Constraint Satisfaction and Scheduling," in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.

[8] H. Kautz, B. Selman, "Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search," *Proceedings AAAI96.*

[9] S. Lin and B. Kernighan, "An Effective Heuristic for the Traveling Salesman Problem," *Operations Research Vol. 21*, 1973.

[10] D. Mittman (mission operations and planning lead for Space Infra-red Telescope (SIRTF) Mission, personal communications, April 1997.

[11] N. Muscettola, B. Smith, S. Chien , C. Fry , K. Rajan, S. Mohan, G. Rabideau , D. Yan, "On-board Planning for the New Millennium Deep Space One Spacecraft," *Proceedings of the 1997 IEEE Aerospace Conference*, Aspen, CO, February, 1997, v. 1, pp. 303-318.

[12] B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, and B. Williams, " An Autonomous Spacecraft Agent Prototype," *Autonomous Robots*, March 1998.

[13] R. Ridenoure, New Millennium Mission Operations Study (and Personal Communication to Guy Man), June 1995.

[14] R. Simmons, "Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems," Technical Report, MIT Artificial Intelligence Laboratory, 1988.

[15] S. Smith, "OPIS: An Architecture and Methodology for Reactive Scheduling," in *Intelligent Scheduling*, Morgan Kaufman.

[16] G. Sussman, "A Computational Model of Skill Acquisition," Technical Report, MIT Artificial Intelligence Laboratory, 1973.

[17] M. Zweben, B. Daun, E. Davis, and M. Deale, "Scheduling and Rescheduling with Iterative Repair," in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.