

Planning and Monitoring Solar Array Operations on the ISS

Sudhakar Y. Reddy¹, Michael J. Iatauro¹, Elif Kürklü¹, Matthew E. Boyce², Jeremy D. Frank³,
and Ari K. Jónsson⁴

¹Perot Systems Government Services, ²MCT, Inc., ³NASA Ames Research Center, ⁴Reykjavik University
NASA Ames Research Center, Moffet Field, CA 94035
{Sudhakar.Y.Reddy, Michael.J.Iatauro, Elif.Kurklu, Matthew.E.Boyce, Jeremy.D.Frank}@nasa.gov, Ari@ru.is

Abstract

Managing the International Space Station (ISS) solar arrays requires flight controllers to constantly balance multiple complex constraints against power needs. The complexity not only impacts planning activities, but has an even more acute effect on real-time operations, in particular when handling unexpected events or changes in operations plans. The Solar Array Constraint Engine (SACE) has been developed to assist the flight controllers with the task of planning and executing solar array operations in a safe and effective manner. SACE is built on top of the EUROPA₂ model-based planning system, using its constraint management and automated planning capabilities to reason about the different constraints, find optimal array modes and orientations subject to these constraints and user-configurable solution preferences, and automatically generate solar array operations plans. In addition to operations planning, SACE provides situational awareness, what-if analysis, and optimization functionality.

Introduction

As the International Space Station (ISS) nears completion, new solar arrays have been added to improve the power availability to meet the demands of the new science and crew modules. At the completion of assembly, the ISS will have eight solar arrays that can be oriented in two dimensions. This is in contrast to a much simpler initial configuration on the ISS, where a single solar panel had only one degree of freedom and was largely out of harm's way. The new solar panels add a great deal of complexity to ISS operations planning and monitoring. The solar arrays are designed to automatically track the sun, as the station revolves around the earth, to maximize the power production. However, normal ISS operations such as water dumps, visiting spacecraft (Space Shuttle, Progress and Soyuz), and extra vehicular activities (EVA) put complex constraints that ultimately define safe array configurations, due to structural load limits, contamination concerns, and thermal impacts, which in turn impacts power generation. Consider, for example, a visit by a Progress spacecraft, which uses thrusters to maneuver around the station. The load from the thrusters on the solar panels is minimized if the arrays are oriented with their edge towards the Progress. In general, due to the risks involved in human flight operations, the operations are very conservative,

requiring extensive precautions and contingency planning, which in turn add more constraints to the problem.

ISS Mission Operations

ISS Mission Operations are complex and involve long preparatory lead times for planning and validating, real-time operations, which can involve nominal and off-nominal situations, upfront and real-time coordination among multiple groups, and more. Operations are controlled by experts in mission control, organized into teams that manage certain aspects or subsystems of the station.

The Power, Heating, and Lighting Controllers (PHALCON) are responsible for planning and monitoring the power, heating, and lighting systems for safe operations. This involves constantly balancing multiple complex constraints against power needs and power production capability. The power system itself consists of the solar arrays, the joints that orient the arrays, the batteries, the charging systems, the power loads, etc. PHALCONs interact closely with other groups, such as the Power Resource Officers (PROs), attitude control experts, and, due to the key role that power plays, eventually with all other groups. The interaction of power control and attitude control is a good example of the complexity of these operations. Even during routine operations, the attitude (roll, pitch, and yaw) of the ISS needs to be changed for docking and undocking, debris avoidance, and re-boosting for orbit correction. PHALCONs must closely coordinate their plans with the attitude control team, because the attitude change is accomplished through firing one or more thrusters attached to the ISS or the docked vehicles. These actions put loads on the solar arrays and subject them to contamination if they are not properly oriented. Attitude and other configuration changes also impact how arrays are best oriented for power generation, which in turn might impact power availability.

Currently, PHALCONs take about four weeks of calendar time to manually produce an ISS solar array operations plan for a typical four week planning horizon, a process requiring manual transfer of information between different teams. Furthermore, certain constraints can only be checked when a plan is fairly complete, requiring multiple revisions to the plan before a valid one that meets all the different constraints can be produced.

AI Planning in ISS Mission Control

At a first glance, it might appear that optimization and automated planning in this domain would be straightforward and easily mapped to standard representation and reasoning approaches. However, the complexity of the problem, mission operations culture, and the nature of AI technology raised a number of interesting and hard technical challenges.

Flight planners and controllers use certified processes and procedures in determining the optimal, but safe, orientations of solar arrays. The processes define “imperative algorithms” for safe operations of the solar panels. One of the technical challenges is in mapping the implicit elements of these imperative algorithms into declarative representations used by AI planning systems. Another challenge is in translating the nonlinear continuous constraints into a representation that can be efficiently reasoned by EUROPA₂’s discrete variable and value-based constraint reasoning engine. The very important, but often overlooked, challenge is in supporting a complex requirements analysis and development process for producing flight-certified software, which inevitably involved changes in the problem being solved and subsequent changes to the solution methods.

The objective of this article is to introduce a very real and interesting problem to the planning community, to describe how AI planning and constraint reasoning technology is being used to solve this real problem, and to give insight into what it takes to work with customers to develop complex applications of planning technology. The paper is organized as follows: first, we define the problem; next we present the approach taken in the tool we developed; then we describe the functionality and interface of the tool; and finally we end with some concluding remarks and notes about future work.

The Problem

As previously stated, ISS will have eight solar arrays, each of them mounted on a rotary joint called the Beta Gimbal Assembly (BGA). Four each of the BGAs are mounted on a Solar Alpha Rotary Joint (SARJ), one each on the starboard and the port sides of the ISS. Therefore, each solar array has two degrees of rotational freedom, though some degrees of freedom are constrained by the shared SARJs. Further, the rotary joints can be in different modes e.g. auto-tracking the sun, parked in a specific position, latched (for BGA) or locked (for SARJ). The objective of the Solar Array Constraint Engine (SACE) is to determine the appropriate modes and the orientations for the different joints for safe operation under different ISS configurations and events. This capability is used both for analysis of current or future ISS configurations, and for producing long-term array plans. The decision variables are the orientations and modes of the different joints, and the control variables are those that determine the configuration. The latter include the attitude (roll, pitch,

and yaw), combination of thrusters firing, the specific event – docking, undocking, attitude hold, water dump, etc. It should be noted that constraints may only be applicable in some joint assignments of the control variables (e.g. array loads conditioned on spacecraft docking).

Solar Array Constraints

Solar array constraints fall into the following categories: power, loading (array and SARJ are treated separately), contamination and longeron shadowing. The power availability due to any array will be the maximum if it automatically tracks the sun, but this is not always a safe mode in which to operate. If power availability drops too low, some ISS subsystems must be shut off. As described earlier, the attitude of the station needs to be changed periodically to account for various events by firing a combination of different thrusters, which in turn imposes structural loads on the solar arrays as well as the joints, especially the SARJs. Further, the thruster plumes and water dumps can cause contamination of the arrays, reducing their power generation.

Additionally, differential shading of longerons, which are structural elements that keep the array blankets in tension, put stresses on the arrays, with the magnitude of the stresses depending on a complex set of calculations. An array’s longerons can be shadowed by its own blanket or those of a neighboring array, the amount of shadowing depending on the solar beta (the elevation of the sun relative to the orbit plane of the ISS), and the orientations of the adjacent arrays. The same factors also impact the power generation by the solar arrays; to improve power, the arrays should not be shadowed, but to keep the arrays from shadowing each other or the longerons, they are no longer in an orientation for producing maximum power.

The various constraints map array configurations into three color-coded zones – red when the constraint is violated and the orientation is considered an infeasible solution or a keep out zone, green when the operating zone is feasible, and yellow an intermediate zone where one can operate if there is no green zone solution available.

Orientation and Mode Optimization

For a given configuration of the ISS, the main problem is to find orientations and modes for the different arrays that keep them in a safe operating zone and at the same time maximize the power availability. One way to solve this problem is to pose it as a (nonlinear) mathematical programming problem (or NLP). The ISS certified operations procedures, however, define this problem in terms of solution preferences, rather than as a classical NLP. The solution preference for orientation determination can be paraphrased as follows:

In finding a solution, first avoid all orientations that cause red power, then avoid red loads, next avoid red longeron shadowing, then avoid yellow loads, after that avoid red environment, and then find a location that maximizes power.

Another imperative procedure encodes the solution preferences for determining the mode of the solar arrays, whether to autotrack, park, lock, or latch the various joints. The modes cannot be independently determined for each solar array, because of the SARJ shared by multiple arrays. For example, a part of the preference procedure can be paraphrased as follows:

In determining a mode, prefer autotrack to park, and park to latch or lock. If the array loads are in the red zone, latch the BGA, and if the SARJ loads are in the red zone, lock the SARJ. If the current orientations are safe, but if there is a possibility of the loads on any joint getting into danger zone during autotracking, avoid autotracking that joint. Further, if there is a possibility of the contamination constraints getting into the danger zone during autotracking, avoid autotracking, expect if operating in a contingency mode.

The solution preferences for mode determination are more complex because of the interdependence of BGA and SARJ mode determinations. Further, restrictions on modes for certain joints and preferences between modes for other joints could be specified by the user at run-time, based on the health of the different BGAs and SARJs.

Solar Array Planning

The planning problem is to build a solar array plan to change array modes and orientations, given planned evolution of configurations and constraints in time. The evolution of configurations is defined by an attitude timeline (ATL) and a thruster timeline (TRTL). The events on these timelines can have fixed or flexible start and/or end times. Each configuration is associated with contingency configuration variable values as well. Together, the main and contingency station configurations (C , C_A) define the context for when the different sets of load, contamination, etc. constraints (X) are active. The ATL defines the flexible time interval over which each configuration is active.

The problem state (S) is composed of the array orientations (α , β), joint modes (m), and the station configurations (C , C_A). The possible set of actions (A) include the actions to change the mode (park, lock, latch, autotrack) of the different BGAs and SARJs, and the turn or slew actions to change the orientation of the arrays. Both the state and the action have an extent in time, defined by the start and end times (t_s , t_e); one or both these times could be flexible. The goal of the planning problem is to find a set of actions, states, and their extents that are consistent, do not violate the constraints (X), and optimal with respect to the solution preferences.

Additional constraints that govern planning include the maximum rates at which the different joints can be slewed or turned, and the minimum durations on time intervals between switching modes or orientations to account for minimum time required for authorizing and issuing commands and monitoring their completion. Further objectives during planning include the minimization of the

turns of the rotary joints, which is preferred to maximizing power availability once sufficient power is available to meet critical needs (power is in the green zone). Another consideration for slewing actions is to minimize the change in direction of rotation of the joints.

In summary, each action has complex constraints, in particular for a duration that depends on the context. This is made more complex by the impact that context dependency has on local instance solutions, as they serve to further restrain possible solutions. The problem is also non-directional, as a globally more optimal solution can be obtained by making an early action less optimal. The planning problem is NP complete, and given the size of the problem at hand, it is impractical to obtain a globally optimal solution through exhaustive search.

The Approach

Several guiding principles have driven the choice of the approach to address the problem. First, the PHALCONS wanted to gain confidence in the approach, so the application had to be developed in stages – first to address monitoring and optimization for a single configuration, and in the next phase planning over a time horizon. Second, we needed to follow already certified procedures as much as possible. We chose a constraint-management based approach for monitoring the arrays during real-time execution, - to ensure that they are operating in a safe manner with respect to the different context-dependent constraints. We used an exhaustive optimization approach, which uses a specially designed cost function that faithfully encodes the certified solution preferences to find optimal modes and orientations for any specific configuration.

Finally, our planning approach uses a model-based planning system to model the domain states and actions. However, instead of solving the problem as a global optimization problem over the entire time horizon, we chose a greedy approach that used an optimizer to find a locally-optimal solution at each stage of the time evolution of the plan. Even though the solution is not globally-optimal, it follows the approach currently used by the PHALCONS, and thus makes it easier to gain their confidence in the solutions produced by the tool.

EUROPA₂: Constraint-based Framework

SACE uses the Extensible Universal Remote Operations Planning Architecture (EUROPA₂) framework for optimization and automated planning. This model-based planning system accepts a declarative description of a class of planning problems consisting of a list of *timelines* (concurrent threads of a plan), a list of *states* that may hold on each timeline over an interval, and *compatibilities* describing the relationships that must hold between timelines in order for a plan to be valid. The EUROPA₂ framework provides an interface that allows programmers to build customized planners that meet the needs of their applications. EUROPA₂ incorporates special purpose

modules for reasoning about time, general constraints, managing timelines, managing applicability of compatibilities, and managing search control heuristics. For a detailed description of the EUROPA₂ framework and the underlying concepts, please refer to (Frank and Jónsson 2003).

EUROPA₂ is highly reconfigurable and easily adaptable to different domains, and it has been employed in a variety of NASA missions (e.g. MAPGEN for the Mars Exploration Rovers (Bresina et al. 2005) MSLICE and PSI for upcoming Mars missions (Aghevli et al., 2006)) and many technology demonstrations. The same capabilities made it suitable for managing the complexities of managing the ISS Solar Arrays. For example:

1. Conditional applicability of constraints. EUROPA₂'s language represents such constraints naturally, whether the configuration is determined from telemetry or specified by the operator.
2. Aggregation of constraint classes. Array configurations safety is aggregated using a "least-safe" rule over all applicable constraints. These rules are naturally expressed using constraints to capture the implications.
3. Support for flight controller's desire to selectively ignore certain constraints. Again, EUROPA₂'s language can be used to incorporate flight-controller specified desire to ignore or incorporate a class of constraints.
4. Evolution of the rules. Over the two years the application has been developed, the set of states and rules has changed considerably. The model-based nature of EUROPA₂ has led to less development of new code compared to application-specific methodologies, since EUROPA₂'s rules language is flexible enough to incorporate many of the desired changes.
5. Evolution of functionality. Again, over the two years the application has been developed, the functions of the tool have expanded. Initially all that was desired was the ability to assess orientations, then planning was desired, then the ability to handle complex adjacent array shadowing constraints, etc. The modularity of EUROPA₂'s code base and API allowed incremental expansion of the application.

Representation of Solar Array Constraints

As each solar array has two degrees of freedom, defined by the SARJ angle (α) and BGA angle (β), the natural representation for modeling the different load, contamination, and power constraints is as a 2-dimensional table, with rows representing different values of α and columns representing different values of β . SACE discretizes the domain for the α and β angles into single degree increments. Each cell in the table represents the numerical value of the constraint, for example, structural load. As discussed earlier, the constraint values are categorized into three zones, and the numerical ranges over which a constraint is considered red, yellow or green is configurable. EUROPA₂'s constraint representation makes it convenient to represent and reason about these tabular constraints. SACE allows the user to restrict the domains

for the various angles; this required special care when the domain extends over the 360° to 0° boundary.

Detailed analysis models have been developed by ISS engineers to estimate the loads, contamination, and power availability in different orientations, for different configurations of the ISS. These analyses are too computationally intensive to run on-line during planning or monitoring; however, they can be performed off-line over a known set of configurations. Such tables are constructed off-line and used in SACE. As some of the constraint data is generated at a coarser granularity, SACE uses numerical interpolation to find the constraint values for the intermediate values of α and β .

One of the tougher challenges was translating the imperative representation of the mode determination procedure into a declarative representation. Especially because of the interdependence of the BGA and SARJ modes, this required splitting the procedure into several declarative constraints, together called the *lock-latch constraint set*.

As discussed earlier, differential shadowing of the longerons causes structural stresses that can be catastrophic to the solar array. At any given point in time, one can assess the time to criticality by a complex function of the time in shadow for the four different longerons that are part of a given solar array. During real-time monitoring, these are easy to compute based on the telemetry feeds. However, during planning or optimization, the calculation of a longeron constraint requires the precise information about the starting state of the shadowing timers, and in addition relies on a detailed simulation of the shadowing for a specific configuration and specific modes and orientations of the arrays, as the station revolves around the earth. Therefore, unlike the load and contamination constraints, the longeron shadowing constraints cannot be evaluated ahead of time for use during planning.

Orientation and Mode Optimization

The orientation and mode optimization problem has been posed as an unconstrained optimization problem. Both the constraints and the solution preferences have been encoded into a cost function (L) that is minimized. The optimization problems for the starboard and port sides of the station are independent. On each side, however, the shared SARJ means that the individual arrays cannot be independently optimized. In an early formulation, when adjacent array shadowing was not considered, the overall cost function could be split into n independent functions, where n is the number of BGAs sharing a SARJ, and is either 2 or 4 depending on the stage of assembly. As tool requirements evolved, the need to account for adjacent array shadowing caused interdependence between the optimal solutions for the arrays.

The cost function for each array is a function of the SARJ and BGA angles, and can be represented as a weighted sum of other cost functions:

$$L_i(\alpha, \beta_i) = w_c L_i^c(\alpha, \beta_i) + w_m L_i^m(\alpha, \beta_i) + w_d L_i^d(\alpha, \beta_i) \\ + w_\theta L_i^\theta(\alpha, \beta_i) + w_p L_i^p(\alpha, \beta_i)$$

The component cost functions refer to the color cost, L^c (due to different constraints being in the red, yellow, or green zones), the mode cost, L^m (due to modes of the BGA and the SARJ), the distance cost, L^d (due to distance between current orientation of arrays and the solution), the direction change cost, L^θ (due to change in direction of trajectory of arrays), and power cost, L^p (due to incremental differences in power within a constrained power zone). For example, the color cost encodes the preference discussed in the problem formulation section. We modeled this as a simple linear program and solved for the color costs that encode these preferences. The weights in the above equation are set such that the user-desired preference order of first optimizing using the constraint color zones, next optimizing the modes, then minimizing the distance, etc. is maintained.

The second stage of the algorithm for determining the optimal orientation is fairly simple. The cost for each orientation ($\alpha, \beta_1, \beta_2 \dots$) is calculated, for each α , as the sum of the costs for the best β_i with respect to the corresponding L_i .

$$L(\alpha, \beta_1, \beta_2, \dots) = L_1(\alpha, \beta_1^*) + L_2(\alpha, \beta_2^*) + \dots$$

The optimal orientation is then the orientation that minimizes the overall cost, L . With adjacent blanket shadowing, however, the power produced depends on the orientations of the pair of arrays. So, the power cost, L^p , cannot be independently estimated for each array. Therefore, the power cost computation is moved to this second-stage of the algorithm. Further, the second stage needs to exhaustively search the ($\alpha, \beta_1, \beta_2, \dots$) space to determine the optimal orientation.

Once an optimal orientation is determined, we then determine the modes for the different joints. The mode costs used in the cost function of the first stage are optimistic lower-bound estimates. Therefore, the full mode determination preference described in the problem formulation section is used to determine the mode.

Automated Solar Array Plan Generation

SACE represents the states and actions, described in the problem formulation, on timelines in EUROPA₂. In EUROPA₂, there is no representative difference between states and actions, but for this discussion, the configurations, array orientations, and joint modes are treated as states. The actions either change the mode (Park, Latch, Lock, Autotrack actions) or change the orientation (Turn/Slew action). The basic planning constraints are: (a) array orientation does not violate feasibility constraints on load, contamination, etc.; (b) array modes are feasible for chosen orientations; (c) joints are in position before parking, latching, or locking; and (d) actions meet minimum duration requirements for commanding and execution.

SACE uses the optimizer discussed in the previous section to address the first two constraints, and uses the EUROPA₂ planner to address the latter two constraints. In essence, it treats the global optimization problem during

solar array planning as simply a sequence of individual-configuration optimization problems. Such a simplifying assumption can lead SACE into a situation where it cannot find a feasible solution for a down-stream configuration even when one exists, because of a greedy choice for an earlier configuration. The means of handling this situation is described in the following paragraphs.

In the initial phase of the three-phase planning process, information from the ATL and the TRTL are represented as *tokens* on respective *timelines*. The event tokens on these timelines are then translated into one or more configuration tokens on a different timeline, based on rules in the planning domain model. The SACE planner then iterates over the configuration tokens, finding the optimal orientation and mode using the local optimizer. Each BGA and SARJ is represented by a timeline as well, and the optimal solution is translated into state tokens (to represent orientation and mode) and action tokens (to change mode or orientation) on these timelines. The constraint management system in EUROPA₂ restricts the domain of orientations the down-stream configurations as the solution for a given configuration is specified on the timeline.

After each configuration token is optimized, the planner validates the resulting orientations and modes generated thus far to detect any conflicts. Problems may arise due to insufficient duration to turn an array from one orientation to the next in between two configurations, insufficient duration for lock/unlock a SARJ array and/or latch/unlatch a BGA, insufficient duration for commanding a turn action, etc. In case an inconsistency is detected, the planner retracts the solution for the previous configuration, merges the two configurations, and repeats the process. To elaborate, suppose configurations are treated in order $C_1 \dots C_k$. If an inconsistency is detected after optimizing configuration C_i , the planner retracts the solution for configurations C_i and C_{i-1} , merges the configurations, and re-optimizes. In our experience, most conflicts are due to insufficient duration for actions and can be resolved by merging configurations.

Once the entire configuration timeline is processed through the optimizer, the resulting BGA and SARJ timelines are processed by the EUROPA₂ planner based on the domain model that defines the latter two constraints in the constraint list described above. The final plan, represented by the BGA and SARJ timelines, is post-processed first to determine feasibility with respect to the longeron shadowing constraint, and next to calculate the detailed power availability along the timeline.

The Tool

The SACE tool has been implemented to assist the ISS PHALCONs with the task of planning and executing solar array operations in a safe and effective manner. To address the three major functions of the PHALCONs, SACE provides a “telemetry view” for situational awareness, a “sandbox view” for what-if analysis and optimization, and a “plan view” for automated planning.

To rapidly demonstrate value, on par with and beyond the other tools developed for PHALCONs, SACE has been developed in stages, using a spiral development process. First, the situational awareness piece was developed; this required the representation of the “table” constraints in the EUROPA₂ framework, as well as translating some of the imperative procedures currently used by the PHALCONs into declarative constraints or systems of constraints. Next the sandbox capability was added, whereby the user can evaluate arbitrary orientations in any given configuration in terms of the various constraints and power availability, or to automatically find an optimal orientation subject to user restrictions on the search space. Finally, the automated planning piece was developed, which required modeling the arrays in the NDDL planning domain description language supported by EUROPA₂, and the implementation of a custom planner.

Throughout this process, as is common in software development, requirements were continually refined, and new requirements were added. For example, metal shavings were discovered in one of the SARJs, requiring that the SARJ be turned as little as possible and be parked or locked otherwise. Due to the constraint- and model-based framework used in SACE, this was fairly straight forward to model, and required minor changes to the user interface. The major reason for the success of this effort is the close coordination between the PHALCONs and the development team through twice-weekly teleconferences in addition to periodic visits to Mission Control to observe the PHALCONs in action.

Architecture and User Interface

SACE is a two-tier application built on top of EUROPA₂ model-based planning system. The back-end provides the constraint propagation, optimization, and planning services, front-end manages the interaction with the user.

Both the telemetry and the sandbox use the same interface components with a few operational differences. In the telemetry view, SACE receives the state information (configuration, orientation, and mode) from the ISS telemetry stream through the ISP interface, whereas in the

sandbox mode, user can input the configuration and other state information. Time-sensitive constraints like the longeron shadowing constraint are computed in real time on the telemetry side, and are taken to be the worst-case over an orbit in the sandbox. Additionally, the sandbox has an interface for invoking the optimizer.

The main window, which has a common form for the telemetry and sandbox, is shown in Figure 1, and has three different areas. The top left area shows the orientations, modes, and over-all summary of the constraint status for the BGAs and SARJs. The top right area shows the configuration variables, which determine the context for the different constraints. SACE reads the constraint information given in the constraint definition files, uses the configuration information in this region to determine the set of applicable constraints, and then evaluates each constraint, providing an indication of whether the state of that constraint is unknown, safe, caution or danger. Per-constraint state information is displayed at the bottom of the main sandbox window.

Contextual information, providing an indication of constraint states for ranges of angles other than the currently chosen ones. The aggregate constraint state for all ranges of angles can be displayed in two primary forms: a map view displays a two dimensional map, showing the aggregate constraint state for any combination of α and β ; a ring view displays a set of concentric rings, showing the constraint state of each individual constraint as well as the aggregate state for all β values given an α and all α values given a β value for each BGA. The map view is shown in Figure 2, and uses color-coding to display the status of the constraints and uses intensity to show power availability.

The planning component of SACE allows the user to load an ATL and a TRTL from files and automatically determines a solar array plan. The resulting plan is displayed as interactive timelines, as shown in Figure 3. The user can review the plan in this window, or dig deeper by loading each configuration into the sandbox by clicking on the desired configuration. The user can also edit the plan. Possible editing options include moving ATL elements along the timeline or adjusting their duration and restrict-

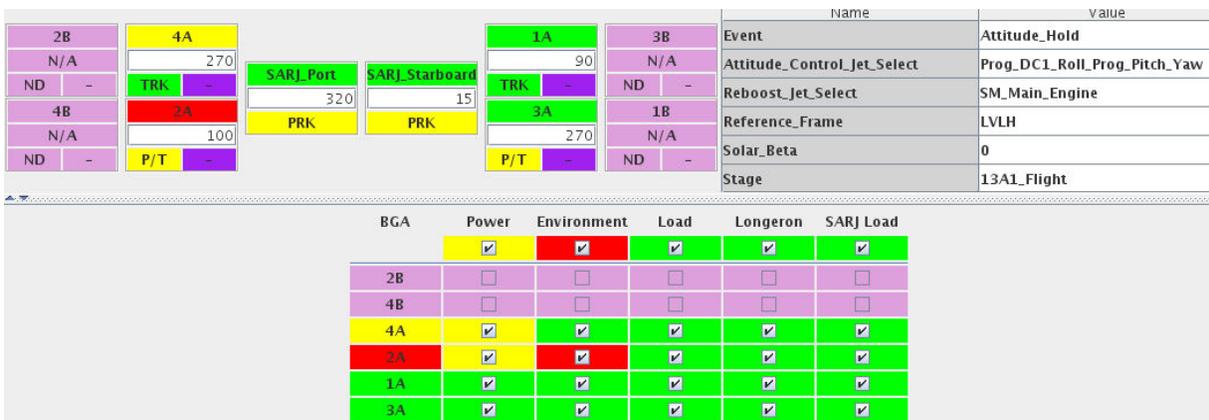


Figure 1. The main window of the SACE Sandbox shows the configuration variables on the top right, the orientations and modes on the top left and a summary view of the constraint status at the bottom.

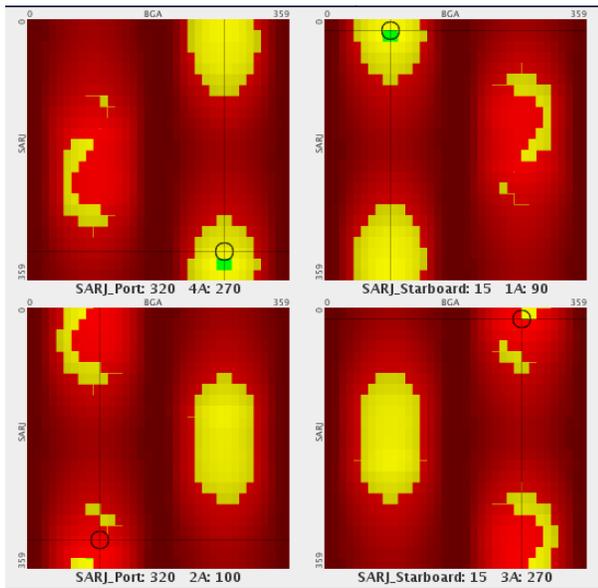


Figure 2. The map view in SACE shows the status of the constraints as a color-coded map, where the Y-axis is α and X-axis is β . The four maps correspond to the four active solar arrays in this configuration.

ing the orientations and modes for the different joints.

Among the elements in the planning window are the SARJ and BGA timelines, showing the modes (including orientations) and turns. Other timelines show the status of the longeron shadowing constraint and the power availability. SACE interacts with a trajectory and power modeling tool called SOLAR for calculating the power availability, the direction of the sun, and the paths taken by the arrays while tracking the sun.

Into Mission Control

Getting automated optimization and planning technology into human spaceflight mission operations is a major achievement. The challenge is to identify a need for the technology in mission control, and then provide significant value that makes up for the technical risk of bringing in

new software, especially one based on AI. In our case, a key application was identified by the ISS mission controllers, for which the automated planning technology seemed to be well suited. The first step was then to build a prototype and demonstrate the capabilities offered by the technology. After that, came a phase of requirements specification and refinements; this in turn led to a decision to commit to the development of the tool.

The development of such a tool is far from simple; even if no new methods need to be developed. Requirements change over time, and constraints that initially were assumed to be simple table constraints turned into complex calculations. During this time, the tool must also be usable by customers, so as to enable evaluation and feedback.

Certification process: The strictest challenge to getting software into mission control is the certification process. For all mission control applications, this boils down to development and testing documentation and a formal approval by responsible parties. Testing out all possible cases and modes is critical to ensure correctness. This is impossible for many software applications, but truly insurmountable for AI-based applications. As a result, the test cases had to be developed as being exemplary, rather than fully covering. The formal approval process is based on the customer team evaluating the tool, running all test cases, as well as being part of the development process to ensure adherence to rules about coding.

Related Work

The problem addressed by SACE is quite different from other problems addressed by automated planning. A few automated planning applications have tackled power management for spacecraft, but none have done it at the level of complexity that SACE does. The first AI system to manage a spacecraft was the Remote Agent (Muscettola, et.al 1998). Part of the agent was the Remote Agent planning system (Jónsson, et.al 2000), which automatically generated plans to achieve operations goals, taking a simple power model into account.

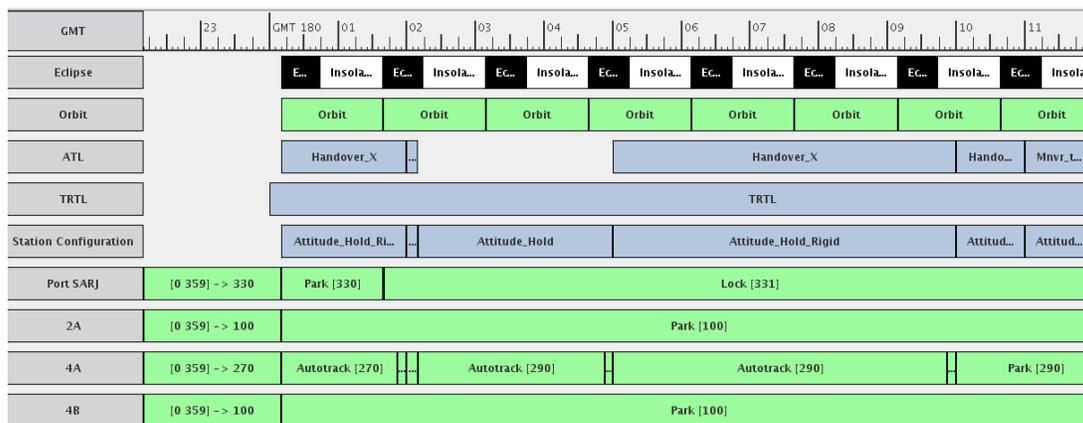


Figure 3. The SACE planning window shows the timelines for the ATL, TRTL, the configurations, and the different SARJs and BGAs, among other things.

The potential of automated on-board power management has been demonstrated for a satellite with signal processing payloads (Shriver, et. al., 2002). The satellite's power system is considerably simpler than that of the ISS, and the spacecraft attitude (and thus power generation) is fixed; however, the onboard system can choose data processing options of differing power consumption for different expected science return. Automated power management was used for NASA Goddard's ST-5 mission (Stanley, et. al., 2005). In this work, the power management system ran in a fully automated mode in Mission Control Center, but its role was limited to characterizing the performance of the ST-5 power system, and notifying mission planners of future constraint violations in the mission schedule due to changed expected power availability or consumption.

The ASPEN model-based planning system from NASA's Jet Propulsion Laboratory was used in a demonstration of automated scheduling of a lower-fidelity model of the ISS power system (Rabideau, 2008). However, this model lacked the highly detailed, context-dependent models required to manage the ISS, and did not manage live updates from the ISS telemetry stream.

Mixed-initiative planning systems have been used in spacecraft operations before. The first was the MAPGEN system, used to build plans for Mars Exploration Rover mission (Bresina et.al., 2005). MAPGEN was also based on the EUROPA planning framework and assisted users with building efficient and safe rover plans each day. The tool, however, did not have a fully integrated power planning element, but rather relied on a simplified power model to do planning and then checked the plan against a more complex calculation. A more recent AI tool in mission operations is the MEXAR2 mixed-initiative planning tool, which is used to plan downlink operations for the Mars Express spacecraft (Cesta et.al, 2007). It does not handle any power planning, but rather focuses on data storage.

Concluding Remarks

We have described the challenging task of automated planning and optimization of ISS solar array operations. The resulting mission operations tool, called SACE, will reduce the time it takes to produce solar array plans from weeks to hours by providing an end-to-end solution that starts with a timeline of ISS attitudes and events and automatically produces a solar array plan and a timeline of power availability. The tool also supports real-time monitoring and responses to unexpected events or changes. SACE is currently undergoing testing for flight certification by the Mission Operations Directorate. Once deployed, SACE will be the first optimization and automated planning tool to be used for human spaceflight.

The optimizer in SACE was carefully crafted to take advantage of the independence of parts of the optimization problem. During the course of the development of SACE and station construction, some of these independence assumptions were invalidated, requiring the modification of the algorithm. This has dramatically increased the time

for optimization of a single configuration, which in turn has increased the planning time to a few hours from about half-an-hour in our first implementations. We plan on exploring approaches to improve the optimization speed.

During planning, SACE solves for a locally optimal solution for each configuration along the timeline. It is possible that a locally optimal solution for one configuration can restrict the space for future configurations so much that a feasible solution cannot be found. Currently, SACE merges locally optimal configurations and makes limited adjustments so as to expand the feasible space. However, such a solution will likely be sub-optimal. Standard search techniques will not work, due to the time it takes to evaluate constraints for each search node, but we plan on exploring variations of backtracking approaches and intelligent search heuristics to address this potential problem.

References

- Rabideau, G. 2008. Space Habitat Power Management with an Autonomous Scheduling System, In *Proc. 9th International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS 2008)*.
- Shriver, P., Gokhale, M., Briles, S., Kang, D., Cai, M., McCabe, K., Crago, S., and Suh, J. 2002. A Power Aware, Satellite-Based Parallel Signal Processing Scheme. *Power Aware Computing*, Kluwer Academic Publishers, Norwell, MA, 243-259.
- Stanley, J., Shendock, R., Witt, K., and Mandl, D. 2005. A Model-Based Approach to Controlling the ST-5 Constellation Lights Out Using the GMSEC Message Bus and Simulink. In *Proc. Intl. Conf. Software Engineering Research and Practice*.
- Frank, J., and Jónsson, A. 2003. Constraint-based Attribute and Interval Planning. *Constraints* 8(4): 339-364.
- Muscettola, N., Nayak, P., Pell, B., and Williams, B. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1/2), August 1998
- Jónsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B. Planning in interplanetary space: Theory and practice. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, 2000.
- Cesta, A., Cortellessa, G., Denis, M., Donati, A., Fratini, S., Oddi, A., Policella, N., Rabenau, E. and Schulster, J. MEXAR2: AI Solves Mission Planner Problems. In *IEEE Intelligent Systems*, 22(4):12-19, 2007
- Bresina, J., Jónsson, A., Morris, P., and Rajan, K. "Activity Planning for Mars Exploration Rovers", in *Proceedings of 15th International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.
- A. Aghevli, A. Bachmann, J. Bresina, K. Greene, B. Kanefsky, J. Kurien, M. McCurdy, P. Morris, G. Pyrzak, C. Ratterman, A. Vera, S. Wragg Planning Applications for Three Mars Missions with Ensemble. *Proceedings of the 5th International Workshop on Planning and Scheduling for Space*, 2006.