Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

Recent Improvements to HST Parallel Scheduling

Ronald Henry, Mike Butschky

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218

Abstract. The Hubble Space Telescope (HST) has several scientific instruments (SIs) that may be used at any given time. Most primary visits submitted by HST observers only use one SI, leaving the other SIs free to be requested by "pure parallel" observing programs. In order to accomplish this, separate scheduling units (SUs) for each parallel SI must be created and then scheduled by the Science Planning and Scheduling System (SPSS), taking into account numerous orbital and scientific constraints. The Parallel Observation Matching System (POMS) has the task of matching parallel visits to primary observations and "crafting" appropriate parallel SUs at each opportunity, taking scientific criteria and orbital constraints into account.

The process for planning and scheduling parallel observations is thus quite different from the process for primary science. In the past, custom crafting rules for each parallel program were necessary, requiring full-time support from a software developer. In addition, because POMS ran as a standalone system, its ability to model how long parallel SUs would take was limited, especially with the flexible buffer-management schemes used for the second-generation SIs.

A new version of POMS was developed in 1997. This version uses a formal proposal syntax (the same used for primary observations) for parallels, so that different proposals can be handled uniformly and without the need for customized "crafting rules." In addition, POMS is integrated with the Transformation (TRANS) planning system in order to give it full knowledge of overheads within an SU, eliminating the need for ad hoc modeling. The power and versatility of this approach has paid off in improved utilization of parallel opportunities, greatly reduced maintenance costs, and an ability to gracefully handle new parallel proposals and new SIs with minimal software effort.

This paper discusses the requirements, design, and operational results of the new POMS.

1. POMS Requirements

POMS has two main sources of input: a description of available parallel proposals, and a description of the primary calendar (this description is generated **after** primary observations are scheduled). Parallel proposals are ranked by priority and structured into visits, which are ranked by priority within each proposal. The task of POMS is to match parallel visits to appropriate intervals on the calendar.

The first step in this process is to structure the calendar into a set of **slots** where parallel alignments (uninterrupted pointings) can go. Oversimplifying somewhat, a slot for a given parallel SI is an interval on the calendar during which there is target visibility, no use of that SI by primary alignments, no spacecraft maneuvers or SAA passages, no loss of pointing control, and no primary alignments marked as non-parallel-able.

For each parallel visit, POMS then structures the slots into a collection of **parallel fields**. Whereas a slot is a place where a parallel alignment can go, a field is a place where an entire parallel SU can go. An SU consists of a set of alignments, but there may be gaps between the alignments (to allow for occultations, for instance), so a parallel SU can extend over many orbits. A field is a set of adjacent slots that (a) all fit within the same primary SU (no major slews within a field), (b) are all scientifically compatible with the parallel visit under consideration (the primary pointing must satisfy any pointing constraints of the parallel), and (c) do not differ in pointing by more than the tolerance specified in the parallel proposal. In order to allow for internal alignments (to do "clean-up" commanding and dump the SI buffers) at the end of a parallel SU, a field is allowed to extend past the end of the last slot in it, until an event is encountered that violates one of the above constraints.

Field sizes provide an upper bound on the size of SUs that can be crafted in them. (In POMS, unlike in primary HST scheduling, there is a difference between the term **visit** and **SU:** a visit refers to the set of exposures submitted by the observer, while the SU is the entity actually scheduled from that visit. Individual visits can be instantiated into parallel SUs many times, whenever POMS finds a suitable opportunity.) Depending on the calendar structure, there will be a distribution of fields of various sizes. The longest opportunities (fields) are considered most valuable scientifically. Normally, parallel proposers are advised to provide a range of visits of various sizes, with the longer ones given higher priority so that POMS will consider them first. If it proves infeasible to craft an SU for a long visit in a field, a shorter visit may be tried in the same place later. If an SU only fills part of a field, the field may be reused for the same visit, as well as for lower-priority visits. Thus the visit sizes act as a sieve, which produces smaller and smaller holes as the shorter visits are encountered. For each SI, at least one low-priority "sponge" visit is provided to fill all the remaining fields that are large enough to allow science that might be conceivably useful.

In order to determine whether crafting is feasible for a particular visit in a field, POMS must compare the field size to the duration of the SU the visit would produce. The duration of an SU includes overheads such as filter wheel motions, cleanup commanding, and buffer dumps. In addition, exposure durations are adjustable within limits provided by the observer. Therefore POMS faces an **overhead inversion** problem. For efficient crafting, it must determine the maximum exposure durations which, after overheads are added, would produce an SU that comes as close as possible to filling the field without exceeding it. However, POMS itself has no knowledge of SI overheads.

This problem was one of the main reasons POMS was redesigned for greater integration with TRANS, the planning subsystem that, among other things, computes SI overheads and also has an algorithm for inverting them to optimize primary observations. The old system attempted to "model" what TRANS would do, an approach that became untenable with the complexities of buffer management that were added to TRANS to support NICMOS and STIS. Instead, POMS now calls TRANS directly each time it attempts crafting. TRANS returns either the longest suitable visit that may be crafted in the field (taking advantage of its ability to adjust exposure durations) or an indication of failure, meaning that even with the minimum exposure durations, the field was still too short for the visit.

One major problem with the crafting approach is performance. Crafting attempts to take advantage of late information – the structure of the primary calendar – in order to optimize efficiency for parallel SUs. The disadvantage is that this creates a real-time process. Adding parallels is only one small step in the larger process of building a calendar for a week's worth of HST science, and normally only a few hours are allotted for it. TRANS was designed as a batch process that does planning for an entire visit, and had never been run in a real-time environment before. Calls to TRANS are computationally expensive. In the POMS environment, a typical TRANS call takes anywhere from 10 seconds to several minutes, depending on the size and complexity of the visit. To cut down on the number of futile TRANS calls, POMS uses a heuristic which identifies cases where crafting is bound to fail and gives up immediately, but there are still several hundred TRANS calls in the course of crafting for a typical calendar. POMS provides an **escape hatch** whereby crafting can be terminated after a specified time limit, even if not yet complete. Performance considerations have limited the number of visits, especially long visits, that can be specified per parallel SI. There are plans to address this problem by partitioning the calendar and distributing the load among many machines.

2. POMS Software Design

POMS is written in Common Lisp using the Common Lisp Object System (CLOS). The main classes in POMS are as follows:

Timeline: the calendar after primary observations are scheduled.

Parallel-Proposal-Set: The set of all parallel proposals POMS is attempting to craft.

Parallel-Proposal: A single parallel proposal in the Parallel-Proposal-Set.

Parallel-Visit: A single parallel visit in a Parallel-Proposal.

SI-Slot-Set: Set of slots computed by POMS for a single parallel SI.

Field-Set: Set of fields computed by POMS for a single parallel visit.

Trans: Interface to the TRANS system.

Crafted-Parallel-Visit-Set: Set of POMS crafted parallel visits.

Crafted-Parallel-Visit: A single POMS crafted parallel visit

The algorithm POMS uses to craft parallels is as follows:

1. POMS-Main creates and initializes the Timeline using the Calendar File.
2. POMS-Main creates and initializes the Parallel-Proposal-Set. Parallel-Proposal-Set creates and initializes all the Parallel-Proposals. Parallel-Proposal creates and initializes Parallel-Visits for each visit in the Parallel-Proposal.
3. POMS-Main creates an empty Crafted-Parallel-Visit-Set.
4. POMS-Main calls Parallel-Proposal-Set's Schedule method to schedule all Parallel-Proposals on the Timeline. Parallel-Proposal-Set calls the Get-Slot-Set method to get the SI-Slot-Set for each parallel-able instrument. Parallel-Proposal-Set calls each Parallel-Proposal's Schedule method in priority order.
5. Parallel-Proposal calls each Parallel-Visit's Schedule method in priority order. Parallel-Visit calls the SI-Slot-Set's Get-Fields method to get the Field-Set from the SI-Slot-Set for this Parallel-Visit's instrument. Parallel-Visit determines if each Field in the Field-Set looks valid for crafting, and if it is, calls Trans's Run method for further verification and crafting.
6. POMS-Main calls Crafted-Parallel-Visit's Write-Report to write the crafting report.

3. Operational Results

The redesigned POMS was installed for operational use at the start of 1998. Prior to this installation, old POMS had been used only for WFPC2 parallels and a set of operational tools based on static (rather than crafted) parallel SUs had been used for NICMOS and STIS. An analysis of parallel performance based on the first four months of operations showed an improvement in total parallel time of about 16%, attributable both to improvements in POMS and to operational improvements during this period. Working under very tight real-time constraints, POMS has crafted about 100 parallel SUs per week, including a substantial number of long SUs (three orbits or more), and is averaging about 48 hours of parallel alignment time per week over the three parallel-able SIs. (Analysis shows that this is still only about 43% of the total potential for parallel time, due to various inefficiencies in the system.) After a few startup glitches, operational performance of POMS has been reliable. An average of 5-10% of parallel SUs produced by POMS have failed to schedule, for reasons that are complex and rooted in the imperfect model of the scheduling system used by POMS and TRANS. Holding the run-time down has been a persistent challenge for the operators.

The main objective of the redesign, however, was not to improve performance but to support the new SIs and reduce maintenance costs. By these measures POMS has been an outstanding success. The integration with TRANS has enabled POMS to incorporate the complex overhead models with the newer SIs, and the formal proposal syntax has greatly reduced the effort needed to support new proposals. POMS no longer requires the support of a full-time developer. Nearly all SI-specific code has been removed from POMS, so that few if any changes will be necessary to support future SIs.