Request-Driven Scheduling for NASA's Deep Space Network

Mark D. Johnston, Daniel Tran, Belinda Arroyo, and Chris Page
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109

Abstract

This paper describes recent work undertaken to increase the level of automated scheduling support available to users of NASA's Deep Space Network (DSN). We have adopted a request-driven approach to DSN scheduling, in contrast to the activity-oriented approach used up to now. We describe some of the key constraints and preferences of the DSN scheduling domain and how we have modeled these as scheduling requests. Algorithms to expand requests into valid resource allocations, and to resolve schedule conflicts and unsatisfied requests, have been developed and incorporated into a distributed system of servers called the DSN Scheduling Engine (DSE). To explore the usability aspects of our approach we have developed a pathfinder graphical user interface that utilizes the DSE. This GUI incorporates several key features to make it easier to work with complex scheduling requests, including progressive revelation of detail, immediate propagation and feedback of implications, and a "meeting calendar" metaphor for repeated patterns of requests. This pathfinder system has been deployed and adopted by one of the JPL DSN scheduling teams, representing an initial validation of our overall approach. The DSE is planned to be a central element of the Service Scheduling Software (S^3) web-based scheduling system now under development for deployment to all DSN users.

Introduction

NASA's Deep Space Network (DSN) provides communications services for planetary exploration missions as well as other missions beyond geostationary, supporting both NASA and international users. It also constitutes a scientific observatory in its own right, conducting radar investigations of the moon and planets, in addition to radio science and radio astronomy. The DSN comprises three antenna complexes in Goldstone, California; Madrid, Spain; and Canberra, Australia. Each complex contains one 70m antenna and several 34m antennas, providing S-, X-, and K-band up and downlink services. The distribution in longitude enables full sky coverage and generally provides some overlap in spacecraft visibility between the complexes. A more detailed discussion of the DSN and its capabilities can be found in (Imbriale 2003).

The process of scheduling the DSN is complex and time-consuming. There is significantly more demand for communications services than can be handled by the available assets. There are numerous constraints on the assets and on the timing of communications supports, due to spacecraft and ground operations rules and preferences. Most DSN users require a firm schedule around which to build spacecraft command sequences, weeks to months in advance. Currently there are several distributed teams who work with missions and other users of the DSN to determine their communications needs, provide these as input to an initial draft schedule, then iterate among themselves and work with the users to resolve conflicts and come up with an integrated schedule. This effort has a goal of a conflict-free schedule by eight weeks ahead of the present, which is rarely met in practice. In addition to asset contention, many other factors such as upcoming launches (and their slips) contribute to the difficulty of building up an extended conflict-free schedule.

There have been a variety of efforts over the years to increase the level of automation in the DSN to support scheduling. Currently, the DSN scheduling process is centered around the Service Preparation Subsystem (SPS) which provides a central database for schedules and for the auxiliary data needed by the DSN to actually operate the antennas and communications equipment (e.g. viewperiods, sequence of event files). The TIGRAS program (Borden, Wang, & Fox 1997) is used for schedule viewing and editing, along with a number of other tools for generating specialized reports and graphics. The current effort to improve scheduling automation is designated the Service Scheduling Subsystem, or S^3, which will be integrated with SPS. There are three primary features of S^3 that are expected to improve the scheduling process:

- *unifying the scheduling software and databases* into a single integrated suite covering realtime out through as much as several years into the future
- *adopting a request-driven approach to scheduling* (as contrasted with the current activity-oriented scheduling)
- *development of a peer-to-peer collaboration environment* for DSN users to view, edit, and negotiate schedule changes and conflict resolutions

In this paper we focus on the second of these major capabilities — request-driven scheduling, and its implications in

terms of a scheduling request specification or "language", and on the scheduling algorithms themselves. We first provide some background on the DSN scheduling problem and the existing scheduling tool suite, and on the rationale for the approach taken by S^3. We then describe the scheduling request specification, which is how DSN users will describe their service requests to the system. These requests are processed by the DSN Scheduling Engine (DSE), which expands schedule requests into tracking passes, integrating them into an overall schedule, seeking to minimize conflicts and request violations. A pathfinder graphical user interface has been developed for creating and editing schedule requests, and integrating them into schedules and minimizing conflicts. This pathfinder version has been deployed in a test configuration for several months, and we describe our experiences to date and lessons learned from this preliminary deployment. We conclude with an overall status summary, and a description of plans for ongoing development.

Background

The driving factors towards increased automation of the DSN come from several directions. The expected increase in the number of missions from NASA and international partners will put more and more pressure on the available DSN resources, a trend which is expected to accelerate in the future. More missions are expected to have higher data volumes and greater link complexities. At the same time, there is a strong desire to reduce operations costs, while increasing reliability and continuing to provide 24h service coverage.

Increased automation support for DSN scheduling has a long history. LR-26 was a customizable heuristic scheduling system for the 26-meter antennas using Lagrangian relaxation and constraint satisfaction search techniques(Bell 1992). Operation Mission Planner (OMP-26) used heuristic search to allocate 26-meter antennas to missions, and linear programming to adjust track durations(Kan, Rosas, & Vu 1996). The Demand Access Network Scheduler (DANS) included all antennas and used a heuristic iterative repair approach(Chien *et al.* 1997). Other investigations into are described in (Fisher *et al.* 1998; Clement & Johnston 2005; Johnston & Clement 2005; Guillaume *et al.* 2007).

The current DSN scheduling software project S^3 is derived from a 2004 resource allocation process working group that analyzed the DSN scheduling process and identified a key set of goals for implementation, listed in the Introduction. One of these goals centers on the basic entities that drive the schedule. In the past, and currently, these are the scheduled communications passes (tracks) or other individual activities that are placed on the schedule. All of the software to create, manage, and report the DSN schedule are built around a representation of the schedule as a collection of activities. The shift to a *request-driven* (sometimes called requirements-driven) approach is a fundamental shift in representation, adding a layer above tracks, such that the predominant control mechanism of users over the schedule is via scheduling requests, rather than the individual scheduled activities. Note that it is not anticipated that individual activities can be bypassed; indeed, all the basic capabilities of activity-oriented scheduling are still required: users need to be able to edit individual activities, for reasons that may not be expressible in the form of scheduling requests. However, the net benefits of a request-driven approach outweigh those of activity-oriented scheduling in several important ways:

- *leveraged effort*: one scheduling request can generate and be used to manage many scheduled activities, and one change to a request can propagate to all activities derived from it; this can significantly reduce the ongoing effort needed to generate the schedule and manage its changes
- *automated continuous schedule validation:* based on the request specification, the schedule can be continuously monitored against constraints and preferences; this can help minimize the effort to ensure that schedule changes, as they invariably occur, will not introduce undetected inconsistencies between requests and activities
- *traceability*: all activities trace to scheduling requests that describe the purpose and intent of the generated activities

The main disadvantage of a request-driven approach is that the request specification language is complex. There are many options and subtleties involved in describing the constraints and preferences on DSN activities, and a sufficiently rich representation of these is necessarily large and complicated. Some of the problems that ensue are:

1) what appears at a high level to be a simple request is often much more involved when practical details are considered, yet all of these details may be needed (even if rarely) to fully describe how and when a particular activity can be scheduled. Users do not want to be bombarded with requests for detail when using the system, but neither will they accept that they cannot make use of all available options.

2) many interdependent options can make it difficult to tell whether a request is feasible: the interactions of time windows with other request parameters can all too easily lead to inconsistencies, which may not show up until late in the scheduling process.

3) failure to accurately represent the correct applicable flexibilities forces schedulers to use workarounds that artificially limit flexibility, thus inhibiting user acceptance of the system. For example, if it is not possible to represent that any one of several choices is acceptable, then the human scheduler must pick one, and the advantages of having the flexibility are lost.

These factors pose a major challenge to a request-driven approach, in that the effort of creating and managing requests, and their consequent benefits in continuous validation of schedule, must be shown to be overall more beneficial than an activity-oriented approach in order to gain user acceptance. In the following section we describe how we have approached the problem of representing DSN scheduling requests, and a later section, how we have addressed the way that users can specify complex options.

DSN Scheduling Requests

DSN scheduling requests specify the services required and their associated constraints and preferences.

Services

Services include use of any of the available capabilities of the DSN, including uplink and downlink services, Doppler and ranging (for spacecraft navigation), as well as more specialized capabilities. The details of a spacecraft's service specification depend on the onboard hardware and software (the frequency band, encoding, etc.). Along with other factors such as radiated power levels and distance from the Earth, these all determine a set of antennas and associated equipment (transmitters, receivers, etc.) that must be scheduled to satisfy the request. However, these assets are not all equally desirable, and so there are preferred choices for antennas and equipment that also need to be considered.

In addition to single antenna/single spacecraft communications, there are a variety of other DSN service types. Some missions need the added sensitivity of more than one antenna at once, and so make use of arrayed downlinks using two or more ground antennas. For navigation data, there are special scenarios (DDOR) involving alternating the received signal between the spacecraft and a nearby quasar, over a baseline that extends over multiple complexes. For Mars missions, there is a capability to communicate with several spacecraft at once (called Multiple Spacecraft Per Aperture, or MSPA): while more than one may be sending down data at once, only one at a time may be uplinking. Another feature of the Mars mission complement is the capability to relay data from surface missions such as the Mars Exploration Rover (MER) rovers, via the Mars orbiting missions such as Mars Odyssey and the Mars Reconnaissance Observer (MRO).

Constraints

Constraints on DSN scheduling requests fall into several broad categories. The most important is timing: users need a certain amount of communications contact time in order to download data and upload new command loads, and for obtaining navigation data. How this time is to be allocated is subject to many options, including whether it must be all in one interval or can be spread over several, and whether and how it is related to external events and to spacecraft visibility. Among these factors are:

reducible whether and by how much the requested time can be reduced to fit in an available opportunity

extensible whether and by how much the requested time can be increased to take advantage of available resources

splittable whether the requested time must be provided in one unbroken track, or can be split into two or more

split duration if splittable, the minimum, maximum, and preferred durations of the split segments; the maximum number of split segments

split segment overlap if the split segments must overlap each other, the minimum, maximum, and preferred duration of the overlaps

split segment gaps if the split segments must be separated, the minimum, maximum, and preferred duration of the gaps

viewperiods periods of visibility of a spacecraft from a ground station, possibly constrained to special limits (rise/set, other elevation limits)

events general time intervals that constrain when tracks may be allocated; examples include:

- day of week, time of day (for accommodating shift schedules, daylight, ...)
- orbit/trajectory event intervals (occultations, maneuvers, surface object direct view to Earth, ...)

Different event intervals may be combined and applied to one request. The included events may have a preference ordering.

A second category of constraint is that of *relationships* among contacts. In some cases, contacts need to be sufficiently separated so that onboard data collection has time to accumulate data but not overfill onboard storage. In other cases, there are command loss timers that are triggered if the time interval between contacts is too long, placing the spacecraft into safemode. During critical periods, it may be required to have continuous communications from more than one antenna at once, so some passes are scheduled as backups for others.

A third category of constraint can be called "distribution" requirements. These cover some extended time span and specify constraints on certain aspects of overall set of activities during that time. Examples include: a certain proportion of 70m contacts; ensuring that navigation passes are spread out roughly evenly between the northern and southern hemisphere complexes; ensure that not all contacts in a week are on the same antenna.

Preferences

In addition to constraints, there are numerous preferences that scheduling users have as to how their activities are to be scheduled. Many would prefer additional time if it is available, while at the same time are able to reduce some contact durations in order to resolve a contentious period on an antenna. There are preferences on gap durations, whether tracks are split or continuous, for tracks to occur during day shift at a particular operations center, and so on. While some of these preferences are implicit, some must be explicit and, if they apply, need to be specified as part of the scheduling request.

Priority

Priority plays a significant role in DSN scheduling, but not the dominating role that it plays in some other systems (e.g. (Calzolari *et al.* 2008)). Critical events (launches, surface landings, planetary orbit insertions) preempt other more routine activities. Other than critical activities, missions have higher priorities during their prime (initial phases) than during their later extended missions. However, higher priority does not automatically mean that resource allocations are assured. Depending on their degree of flexibility, missions trade off and compromise in order to meet their own requirements, while attempting to accommodate the requirements of other users. As noted above, one of the key goals of S^3 is to facilitate this process of collaborative scheduling.

Patterns of Requests

One characteristic of DSN scheduling is that, for most users, it is common to have repeated patterns of requests over ex-

tended time intervals. Frequently these intervals correspond to explicit phases of the mission (cruise, approach, fly-by, orbital operations). These patterns can be quite involved, since they interleave communication and navigation requirements. The presence of repeated patterns can be exploited in representing scheduling requests that vary minimally or not at all over some time frame, as will be discussed further below.

DSN Scheduling Engine

The DSN Scheduling Engine (DSE) is that component of S^3 responsible for:

- expanding scheduling requests into individual communications passes by allocating time and resources to each
- identifying conflicts in the schedule, both for resources and for any other violations of DSN scheduling rules, and attempting to find conflict-free allocations
- checking scheduling requests for satisfaction, and attempting to find satisfying solutions

Schedule *conflicts* are based on the schedule alone, not on any correspondence to schedule requests, and indicate either a resource overload (e.g. too many activities scheduled on the available resources) or some other violation of a schedule feasibility rule. Representative examples of conflicts include:

Spacecraft mulitple tracks of the same mission share the same temporal extent

Beginning of Track – BOT Multiple tracks start with in 15 minutes of Goldstone and 30 minutes for Canberra and Madrid.

Start of Activity – SOA Multiple tracks start with in 15 minutes of Goldstone and 30 minutes for Canberra and Madrid.

Antenna (Facility) Multiple non-MSPA tracks use the same antenna at one time

Equipment Multiple tracks share the same equipment during the same temporal extent

Viewperiod The spacecraft/user is out of view of the track antenna

Teardown The post-track teardown time does not match the expected teardown time

Setup The pre-track setup time does not match the expected setup time

In contrast, *violations* are associated with schedule requests and with their tracks, and indicate that the request is not being satisfied in some version of the schedule. Examples of violations include:

Track Quantization The track start or end time violates the request quantization constraint. For example, requests can specify that tracks start or end only at 5 minute intervals.

Track Separation If the request is splittable, the separation time between two tracks violates the split segment overlap or split segment gap constraint.

Track Duration If the request is splittable, the track duration violates the request split duration constraint.

Service Specification The track violates the request service specification, i.e. the antenna or equipment allocated does not match the requested service.

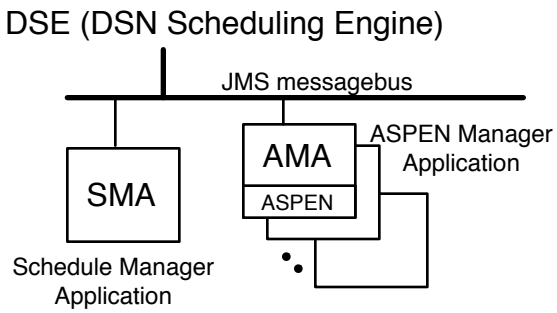


Figure 1: DSE architecture

Total Track Duration The total track duration does not meet the requested duration

Number of Tracks The number of tracks for the requests violates the maximum. For a non-splittable track, this limit is 1; for a splittable track, the limit may be specified.

Track Temporal Extent The track start or end time falls outside the scheduling request's time interval.

Event Reference The track time interval violates the intersection of the event time intervals referenced by the scheduling request.

Request Reference The track time interval violates the scheduling request's temporal constraint link to other requests.

Architecture

The DSE is based on ASPEN, the planning and scheduling framework developed at JPL and previously applied to numerous problem domains (Chien *et al.* 2000). In the context of S^3, there may be many simultaneous users, each working with a different time segment or different private subset of the overall schedule. This has led us to develop an enveloping distributed architecture (Figure 1) with multiple running instances of ASPEN, each available to serve a single user at a time. We use a Java Messaging System (JMS) middleware tier to link the ASPEN instances to their clients, via an ASPEN Manager Application (AMA) associated with each running ASPEN process. A Scheduling Manager Application (SMA) acts as a central registry of available instances and allocates incoming work to free servers. This architecture provides for flexibility and scalability: additional scheduler instances can be brought online simply by starting them and having them register with the SMA.

The DSE communicates with clients using an XML-based messaging protocol, similar to HTTP sessions but with responses to time-consuming operations returned asynchronously. Each active user has a session (possibly more than one) which has loaded all the data related to a schedule that user is working on. This speeds the client-server interaction, especially when editing scheduling requests and activities, when there can be numerous incremental schedule changes.

There are a few basic design principles around which the DSE has been developed, derived from its role as provider of intelligent decision support to DSN schedulers:

- no unexpected schedule changes:

  - all changes to schedule must be requested, explicitly or implicitly
  - the same sequence of operations on the same data will always generate the same schedule
- even for infeasible scheduling requests, attempt to return something "reasonable" in response, possibly by relaxing aspects of the request; along with a diagnosis of the sources of infeasibility, this provides a starting point for users to handle the problem

Algorithms

With these design principles in mind, several automated scheduling algorithms were developed to generate activities from scheduling requests. Users may lock requests and activities to ensure that they are not modified, and the execution of these algorithms is under the explicit control of the user (see GUI description). Also, there are no stochastic elements to these algorithms, thus ensuring that repeated operations with the same data always generate the same schedule.

Initial generation of tracks The initial layout algorithm (Algorithm 1) is executed to initially generate tracks to satisfy the specifications of the request. The algorithm consists of a series of systematic search stages over the legal track intervals, successively relaxing constraints each stage if no solution is found. The systematic search algorithm is a depth-first search algorithm over the space of available antenna start times and durations for each scheduling request. The set of legal antennas for scheduling is defined in the request service specification, while the available start times and durations search space is defined by the request quantization value.

We are employing four relaxation strategies. These strategies are outlined below, with each relaxation strategy building upon the previous.

- temporal linkage — the explicit temporal relationships between tracks in the same or different requests
- track separation — between two track segments from a splittable request
- event intervals — the time intervals (exclusive of viewperiods) that constrain the timing of the track
- spacecraft, antenna, and equipment — removing these conflicts from consideration leaves only the viewperiod constraint

These relaxation strategies allow for tracks to be generated even though the scheduling request may be infeasible (in isolation or within the context of the current schedule), and provides the user a starting point to make any corrective changes as needed. These changes may range from modifying the scheduling request to introduce more tracking flexibility, to contacting other mission schedulers to negotiate different request time opportunities.

Repairing the schedule Once an initial schedule has been generated, conflicts and/or violations may exist in the schedule due to the relaxation of constraints. The DSE provides a basic repair algorithm to reduce conflicts or violations, described as Algorithm 2. Note that conflicts and violations are

Algorithm 1 Initial Layout

```
For each request in the schedule
    Remove existing request tracks
    Systematically search legal intervals to satisfy the request
    If success
        Continue to next request
    End if

    Remove all lower priority tracks in request interval
    Systematically search legal intervals to satisfy the request
    If success
        Add all tracks removed
        Continue to next request
    End if

    Remove all equal priority tracks in request interval
    Systematically search legal intervals to satisfy the request
    If success
        Add all tracks removed
        Continue to next request
    End if

    Remove all remaining tracks in request interval
    Systematically search legal intervals to satisfy the request
    If success
        Add all tracks removed
        Continue to next request
    End if

    For each relaxation strategy
        Systematically search legal intervals to satisfy request
        If success
            Add all tracks removed
            Continue to next request
        End if
    End for
End for
```

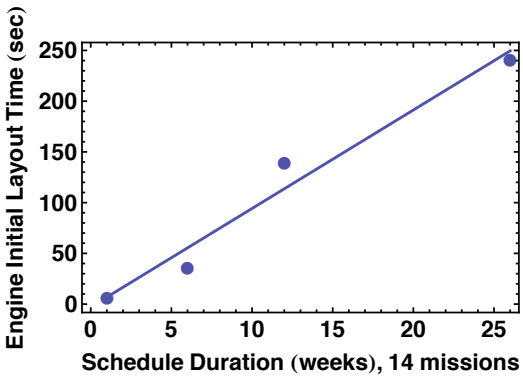
Algorithm 2 Repair Conflicts/Violations

```
Until timeout or schedule is conflict/violation free
    Choose a conflict or violation
    Identify the contributing requests
    For each request
        Checkpoint current state
        Systematically search the legal intervals to satisfy
          the request
        If success or timeout
            Continue to next conflict/violation
        Else
            Recover to checkpoint state
        End if
    End for
End until
```

Algorithm 3 Extend Track To Preferred Duration

```
For each conflict-free track in the schedule
    Checkpoint current state
    Extend duration of track to min(legal interval duration,
      preferred duration requested)
    If violations created
        Recover to checkpoint state
    End if
End for
```

(a)



(b)

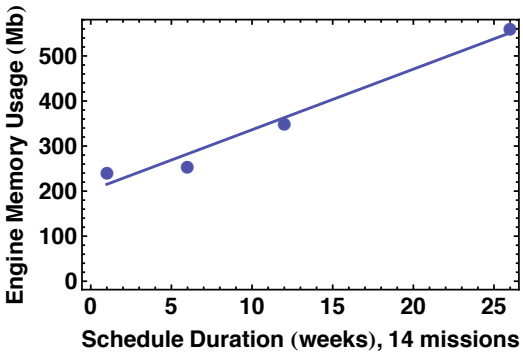


Figure 2: DSE performance scalability for schedules from 1-week to 6-months in duration: (a) run time for initial layout (~10 sec/week) and (b) memory usage (~15Mb/week)

independent, so there are separate versions provided through the user interface for users to invoke.

Optimizing existing tracks in the schedule We also provide the user a method for optimizing existing tracks in the schedule. For requests that are reducible in duration, the above scheduling algorithms may return tracks that, while strictly satisfying the request specifications, have durations that are less than the preferred value, e.g. in order to fit into an available opportunity window. We thus provide an additional algorithm (3) that attempts to achieve the preferred track duration values.

Performance

We have conducted initial performance testing of the DSE, based on schedules of varying duration from 1 week through 6 months. For these tests we used the same 14 mission sample, and repeated their requests uniformly over the entire schedule period. The results are shown in Figure 2: both runtime and memory usage are very well behaved, showing roughly linear growth over the time range of interest.

User Interface

To investigate the capability of the request specification language outlined above, we have developed a pathfinder graphical user interface and web application. The user interface incorporates all of the major basic features of scheduling requests, including viewperiod and event management, and scheduling request creation and editing with all of the features noted above. This UI acted as a DSE client for expanding schedule requests to tracks, identifying and resolving conflicts, and identifying and resolving request violations. The main simplification was to limit the DSE/UI to single-mission, single antenna scenarios, a restriction which is being lifted as further development takes place.

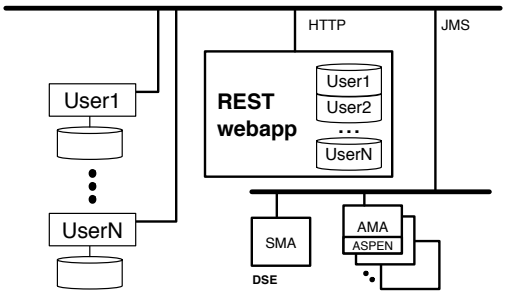


Figure 3: The architecture of the DSE/UI pathfinder user interface

The overall architecture of the DSE+UI is illustrated in Figure 3. Multiple users can work with the system at once, each on their own workstation. Each user has installed a locally running copy of the GUI client, which stores a local copy of all the data needed for scheduling including viewperiod files, event definitions, scheduling requests, and schedules. All changes to these data items are mirrored on a REST-based web application, which also ensures that assigned identifiers are globally unique. Users can then share data items via a command to the web application that transfers over all data associated with a given schedule, including the scheduling requests and any data needed to properly interpret them. This enables users to work on different missions completely independently, yet integrate their requests into a single schedule at the appropriate time. Note that this architecture differs from that of S^3, which is based on a central database and web browser-based client.

The pathfinder GUI was intended to explore and assess several aspects of user interaction with the scheduler:

1. *Progressive revelation of detail*: as noted above, scheduling requests can potentially contain many adjustable parameters, often with interrelationships among them. The GUI uses an animation technique to fade in or out relevant parameter choices, as soon as a dependent choice is made. For example, if a request is for tracking time that is not splittable, then none of the parameters that control splitting are visible on the screen (split minimum duration, maximum number of segments, whether split segments must overlap or be separated, etc.) However, as soon as the user selects the splittable option, a subset of these parameters will fade in. This is chained several levels deep, e.g. overlap parameters settings are not shown unless the user specifies that the split segments must overlap.
2. *Immediate display of implications*: another aspect of the potential complexity of scheduling requests is that it is

easy to overspecify a request, thus making it impossible to satisfy. The pathfinder GUI application adopts a strategy of 1) propagating known information as far as possible, to diagnose problems early, and 2) visually displaying this propagated information. For example, as the user edits a scheduling request, the system dynamically calculates the intersections of viewperiods and all timing event windows, displays the result for all allowable antennas that could potentially satisfy a request, and then checks to see whether the total requested time is available, as well as whether the time requested for any segment is consistent with the request's timing parameters. The results are displayed as a "preview" Gantt view along side the request parameters.

3. *The "meeting calendar" metaphor for repeated patterns of requests*: as noted above, many users formulate their requests as a repeated pattern, with variations. We adopted the metaphor of a meeting calendar program, with which most users are familiar, e.g. in which a meeting or appointment is created and then designated as "recurrent". For DSN scheduling, the repetition intervals are sometimes along typical calendar lines (e.g. daily, weekly), but often are based on trajectory or celestial events (e.g. every visibility interval, or opportunity for a Mars rover to reach earth with its antenna). Additional requirements include the option to place time linkages between successive repetitions, e.g. to prevent two neighboring passes from being too close together.

Once scheduling requests have been created, they may be combined to generate a schedule by invoking the DSE to expand the requirements into explicit tracks. The DSE generates and returns the scheduled activities, identifies conflicts, and checks that all requests are satisfied. The user may invoke a conflict repair strategy, or requirement violation repair strategy, based on the heuristics described above. The GUI allows the user to view the schedule, identify conflicts (shown as red in the Gantt chart view), and see any unsatisfied requests (indicated by a red "$\times$" in the request list on the left). Individual schedule items can be edited, and requests may be locked (fixed in place) and will not be subsequently changed by the DSE. An example of the schedule view is shown in Figure 4.

Pathfinder Deployment

In December 2008 we began a trial deployment to assess how well the concepts described above would work when exercised in a realistic scheduling context. The JPL Multimission Resource Scheduling Services (MRSS) team is responsible for DSN scheduling for 20 different missions (out of about 35 currently being actively scheduled to use DSN resources). One team member started out using the software, and based on positive feedback, the team deployed it in February 2009 to each member. In its current usage mode, each team member develops a set of scheduling requests for their responsible subset of the overall set of MRSS missions. These requests are then integrated by one team member, who prepares an integrated schedule containing all missions for which MRSS is responsible, for delivery to another organization to add additional missions.

Before (manual schedule development)	*After (using request-driven DSE)*
integrated schedule contained only the Mars missions, Cassini, and Spitzer Space Telescope	all 20 MRSS missions are integrated into the schedule
only DSN maintenance and downtime and critical activities were considered when building the integrated schedule	same, with the addition of any other missions for which requirements are available
schedule was developed manually, entered via Excel macro	schedule requests are created and stored, and repeated and re-used from week to week

Table 1: Comparison of before and after process based on MRSS use of the DSE software

The MRSS team's experience with the DSE and pathfinder GUI has been very positive — the most compelling endorsement is that the team does not want to consider falling back to the mode of operations before the software was available. A comparison of the before and after process is provided in Table 1. Among the positive features are:

- repeated requests, and the ability to rapidly "clone" existing requests and edit them to create variations
- the immediate preview capability, providing instant feedback even for complex interval timings
- the ability to quickly create day-of-week based event intervals to constrain scheduling

As of mid-March, the MRSS team has built and delivered 14 weeks of DSN schedule using the DSE test client. The main shortcomings that have been identified center on the simplifications noted above — there is not yet support for multi-user, multi-antennas scheduling scenarios, which still require significant manual intervention. Since the DSE pathfinder GUI does not have extensive scheduling editing capabilities, the DSE schedule is imported into TIGRAS for a final set of interactive updates before delivery.

Future Work

The initial experience with the DSE has been positive, confirming most of the expectations that a combination of an intelligent user interface, combined with user-focused scheduling algorithms and processing, can make a request-driven approach feasible. We plan to continue to use the pathfinder GUI to explore ways to provide more efficiencies to users of the system, including additional preview functionality. Some of the future capabilities to incorporate include:

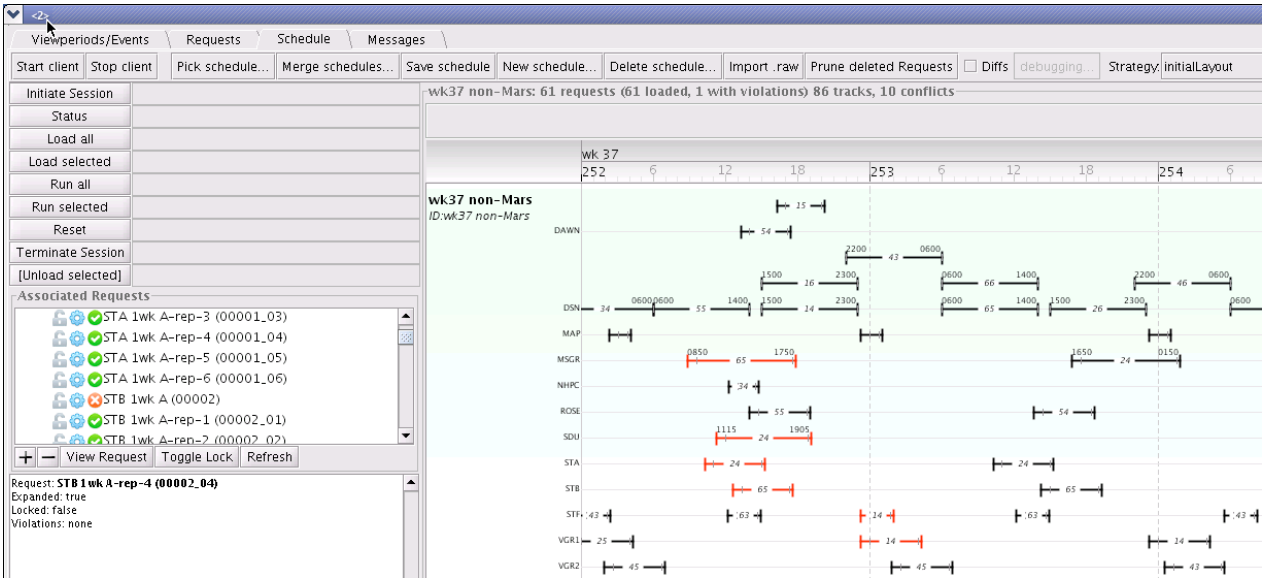


Figure 4: The schedule view showing expanded requests (the list of the left) into tracks (visible in the Gantt view)

- "distribution" requirements, where it is important to apply some global criteria to the track expansion, for example to meet conditions like "during any given month, no more than 75% of a particular mission's tracks should be in one hemisphere"
- ways to define and manage more complex flexibility options, e.g. when there are dependencies among the choices and it is not sufficient to simply provide sets of parameters with value ranges
- support for exploring non-local tradeoffs, e.g. when it is acceptable to reduce a track in one week, but only if it can be restored to requested duration in a preceding or following week

The DSE is a central element of the Service Scheduling Software (S^3) system that is currently in development. When complete, S^3 will provide collaboration and reporting tools as well as scheduling tools, via a web-base architecture that will make it available to all DSN users. The close collaboration between end users and developers has been a key factor in the progress made to date, and we expect this to continue to be critically important as the system development progresses.

References

Bell, C. 1992. Scheduling deep space network data transmissions: A lagrangian relaxation approach. Technical report, Jet Propulsion Laboratory.

Borden, C.; Wang, Y.-F.; and Fox, G. 1997. Planning and scheduling user services for NASA's deep space network. In *1997 Int. Conf. on Planning and Scheduling for Space Expl.*

Calzolari, G.; Beck, T.; Doat, Y.; Unal, M.; Dreihahn, H.; and Niezette, M. 2008. From the EMS concept to operations: First usage of automated planning and scheduling at ESOC. In *SpaceOps 2008*.

Chien, S.; Hill, R.W., J.; Govindjee, A.; Wang, X.; Estlin, T.; Griesel, M.; Lam, R.; and Fayyad, K. 1997. A hierarchical architecture for resource allocation, plan execution, and revision for operation of a network of communications antennas. In *Proceedings IEEE International Conference on Robotics and Automation*.

Chien, S.; Rabideau, G.; Knight, R.; Sherwood, R.; Engelhardt, B.; Mutz, D.; Estlin, T.; Smith, B.; Fisher, F.; Barrett, T.; Stebbins, G.; and Tran, D. 2000. ASPEN - automating space mission operations using automated planning and scheduling. In *SpaceOps 2000*.

Clement, B. J., and Johnston, M. D. 2005. The deep space network scheduling problem. In *Innovative Applications of Artificial Intelligence (IAAI)*. Pittsburgh, PA: AAAI Press.

Fisher, F.; Chien, S.; Paal, L.; Law, E.; Golshan, N.; and Stockett, M. 1998. An automated deep space communications station. In *Proceedings IEEE Aerospace Conference*.

Guillaume, A.; Lee, S.; Wang, Y.; Zheng, H.; Hovden, R.; Chau, S.; Tung, Y.; and Terrile, R. 2007. Deep space network scheduling using evolutionary computational methods. In *2007 IEEE Aerospace Conference*, 1–6.

Imbriale, W. A. 2003. *Large Antennas of the Deep Space Network*. Wiley.

Johnston, M. D., and Clement, B. J. 2005. Automating deep space network scheduling and conflict resolution. In *ISAIRAS-05*.

Kan, E.; Rosas, J.; and Vu, Q. 1996. Operations mission planner - 26m user guide modified 1.0. Technical report, Jet Propulsion Laboratory.