AUTOMATING DEEP SPACE NETWORK SCHEDULING AND CONFLICT RESOLUTION

Mark D. Johnston and Bradley J. Clement

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109 USA
{mark.johnston, bclement}@jpl.nasa.gov

ABSTRACT

The Deep Space Network (DSN) is a central part of NASA's infrastructure for communicating with active space missions, from earth orbit to beyond the solar system. Consisting of more than a dozen major ground antennas at three sites spaced around the globe, it must be carefully scheduled to satisfy the requirements of the various mission users, subject to many constraints. We describe our recent work in modeling the complexities of user requirements, and then scheduling and resolving conflicts on that basis. We emphasize our innovative use of background "intelligent assistants" that carry out search asynchronously while the user is focusing on various aspects of the schedule. These assistants can provide guidance to the user about feasible and optimal solutions to the problem they are working on.

1. INTRODUCTION

The NASA Deep Space Network (DSN) is an international network of antennas that supports interplanetary spacecraft missions and radio and radar astronomy observations for the exploration of the solar system and the universe. The network also supports selected Earth-orbiting missions. The DSN consists of three deep-space communications facilities placed approximately 120 degrees apart around the world: at Goldstone, in California's Mojave Desert; near Madrid, Spain; and near Canberra, Australia. This geographic placement permits constant observation of spacecraft as the Earth rotates.

Each of the DSN sites supports antennas of sizes 26m, 34m, and 70m, along with a variety of receivers and other equipment that can be configured to support all the different kinds of communications requests that must be handled. Fig. 1 shows a picture of one of the largest antennas, the 70m antenna at Goldstone, California, along with one of the 34m antennas at this same complex.

The missions supported by the DSN include all of NASA's deep space missions, such as the planetary exploration spacecraft at Mars (the Mars Exploration Rovers Spirit and Opportunity, Mars Odyssey, Mars Global Surveyor) and Saturn (Cassini-Huygens); cometary explorers such as Deep Impact and Stardust; and missions such as Voyagers 1 and 2 that are leaving the solar system entirely. Other missions supported include observatories such as Chandra (X-ray) and Spitzer (Infrared), in Earth orbit or near Earth orbit. A total of about 20 spacecraft are currently allocated resources in a typical near-term scheduling period, while at the writing of this paper, 61 missions are being scheduled over the ~10 year horizon.

Fig. 1: The 70m antenna at the Deep Space Network Goldstone complex in the California Mojave desert. At the right can be seen one of the 34m antennas.

In this paper we will describe the nature of the DSN scheduling problems, i.e. the allocation of DSN resources to the missions that request them. We will discuss our progress in modeling the requirements of the missions, and in generating schedules that meet these requirements. We will discuss the process of conflict resolution and our approach to automating what is currently a manual and iterative process. Key elements of our approach include tools for visualizing the schedule in various ways, and "intelligent assistants" that can asynchronously examine the schedule and search for possible solutions to conflicts that it contains. We conclude with a discussion of future research and implementation directions.

2. THE DEEP SPACE NETWORK SCHEDULING PROBLEM

Currently, DSN schedules are manually generated out one year in the future. The nearer term portion of the schedule (up to 8 weeks out) considers the specific equipment and personnel to be allocated to a communications event. Beyond this interval, the schedule is concerned primarily with scheduling *tracks*, each of which is an allocation of an antenna to a mission over a specific time period. Tracks are typically one to eight hours in duration and must be placed within a *viewperiod* for that particular mission, i.e. when the spacecraft is visible (within specified geometric limits) from that antenna. A typical week will contain about 370 scheduled tracks. In this paper we are concerned with mid-range (8-week to 6-month) and long-term (beyond 6-months) schedules.

2.1 Constraints

Among the constraints that come into play when generating the DSN schedule are the following:

- A spacecraft can only be scheduled to use an antenna when it is within view: this is primarily a geometric constraint and can be pre-computed ahead of time into a set of usable viewperiods for each mission/antenna pair
- No two spacecraft can use the same antenna at the same time, with the exception that multiple spacecraft in view can downlink at the same time, but only one can uplink. Thus, for example, the multiple missions currently at Mars can share antennas for some activities.
- Tracks have *setup* and *teardown* durations that typically range from 10 minutes to an hour. (The combined setup, track and teardown is called an *activity*.) These setups and teardowns can be scheduled outside of the spacecraft viewperiod but generally cannot overlap other activities on the same antenna.

2.1 Requirements

The missions that use the DSN initially specify their requirements in high-level terms of hours/week per antenna, with more detailed specifications communicated informally (verbally or email) by the missions to the schedulers. One of the challenges faced in automating the DSN scheduling problem is that of defining a representation that can capture the details and subtleties of the individual mission's requirements. The representation must be highly expressive, since mission requirements can incorporate any of the following elements:

- *And/Or* groupings and alternatives. Missions must be able to specify conjunctions of requirements (such as multiple antennas at the same or different times). Such a conjunction means that all of the subrequirements must be satisfied at the same time in order for the requirement to be satisfied. In addition, disjunctive requirements may be specified, such that at most one requirement of a set of alternatives must be satisfied. In general, these requirements can be organized into an AND/OR tree, which is an ideal way to capture the combination of groupings (and) and alternatives (or) that arise when specifying a mission's requirements in detail.
- *Timing*. Tracks may have specified start times and durations, both of which may have allowable ranges.
- *Antennas*. Missions may specify one or more alternative antennas on which a track may be scheduled. They may specify groups of antennas, any one of which is acceptable for a particular type of track.
- *Repetitions*. Track repetitions may be specified in general as requirements, over some time period, on the number of tracks, their durations, overlap, and time gaps to neighboring tracks. All of these quantities may have ranges on them. So, for example, a mission might specify repetition requirements such as "seven tracks/week, of duration 5 to 8 hours with no gap longer than 20 hours between adjacent tracks".
- *Overrides*. In some cases, missions need to override a general requirement with a more specific one based on knowledge of ongoing activities. For example, a general requirement for a telemetry downlink every 10 hours may be superceded around the time of a maneuver with a requirement for continuous coverage.
- *Activity start and end times* for most missions must be scheduled on five-minute boundaries.

There have been a number of previous investigations of DSN scheduling automation. The Operation Mission Planner (OMP-26) used heuristic search to allocate 26m antennas to missions and linear programming to adjust track time intervals [1]. Other automation projects were research efforts and were never deployed. LR-26 was a customisable heuristic scheduler that utilized Lagrangian relaxation [2], while the Demand Access Network Scheduler (DANS) employed an iterative repair technique [3]. Other graphical planning tools that have been used for forecasting, analysis, and interactive scheduling include TIGRAS [4] and FASTER [5]. Our approach combines some of the strengths of these systems, as well as incorporating innovative elements in both scheduling search and in schedule visualization.

Other scheduling domains pose problems that have similar characteristics to the DSN mid- and long-term scheduling problem. The Air Force Satellite Control Network (AFSCN) also schedules large numbers of satellites and ground stations but is limited to one day at a time. For this oversubscribed problem, there appear to be significant "plateaus" in the search space, that make strategies that take larger search steps more effective [6, 7]. Another related problem is that of fleets of Earth observing satellites with onboard resource limitations as well as tight viewperiod constraints [8]. In these problems, where the requests are prioritized, the goal of finding a best subset to fit on a schedule has been addressed by a greedy approach with texture-based heuristics [9].

Automated sequence generation for spacecraft is different in that the horizon of interest is smaller (e.g. one week for an orbiter, one day for a rover) and the model of the spacecraft is very intricate. Safety is the highest priority in sequence scheduling, but efficiency is the goal of DSN resource allocation. Thus, the scheduling problem is very different.

3. AUTOMATING DSN SCHEDULING

Our approach to automating the DSN scheduling problem is illustrated in the architecture diagram of Fig. 2. At the top are shown schematically the DSN data sources such as viewperiods and current schedules in various stages of completion and conflict resolution. This information is available via web services and network databases.

The Interactive User Environment, depicted in the lower left of Fig. 2, consists of several interacting components, including a graphical requirements editor, a timeline display function, and intelligent assistants that provide information to the user about possible scheduling decisions.

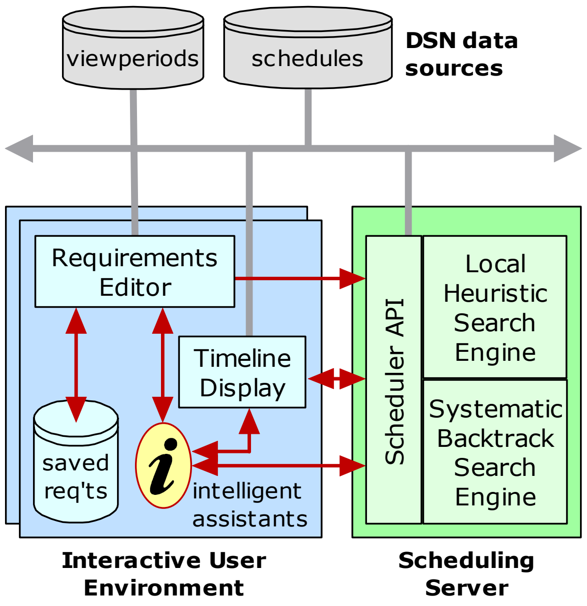


Fig. 2. Architectural overview of automated DSN scheduling

The Scheduling Server, shown in the lower right of Fig. 2, incorporates two major approaches to schedule generation and repair: a local heuristic search technique and a set of systematic backtrack algorithms.

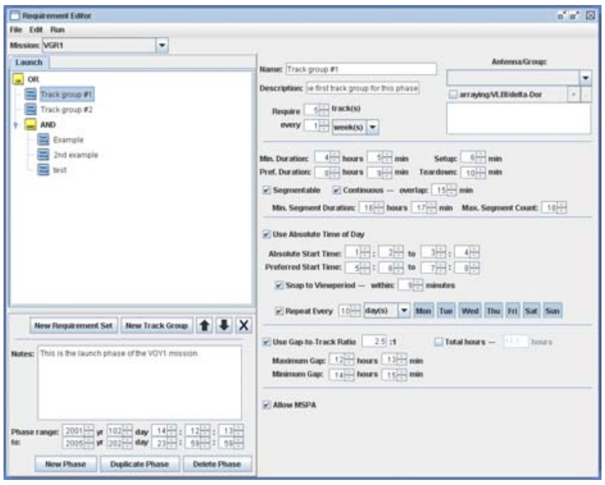
In the following we describe each of these elements in more detail.

3.1 Interactive User Environment

The variety and complexity of the requirements described above provide a challenge for users. To help create and modify mission requirements, we have developed a graphical requirements editor that handles the full expressiveness of the requirements specification for the DSN. Fig. 3 shows a screen snapshot of this editor: on the left is a tree that represents the AND/OR tree of requirements and sub-requirements. As a node is selected, all of its attributes are displayed on the right side of the panel where they may be edited by the user, including the selection of antennas, timing, segmentation, and repetition. Requirements may be saved for further editing, or sent on to the scheduler as described below.

The timeline display provides the user with visual insight into the evolving schedule. We have adapted a standard Gantt-chart view and added additional features to make it easier to navigate and drill down for more information. These features include:

- activity start and end times, as well as setup and teardown times indicated by faint gray bars



Mission: VGR1

Name: Track group #1

Description: e first track group for this phase

Require 5 track(s) every 1 week(s)

Min. Duration: 4 hours 5 min Setup: 6 min

Pref. Duration: 8 hours 9 min Teardown: 10 min

[✓] Segmentable [✓] Continuous — overlap: 15 min

Min. Segment Duration: 16 hours 17 min Max. Segment Count: 18

[✓] Use Absolute Time of Day

Absolute Start Time: 1 : 2 to 3 : 4

Preferred Start Time: 5 : 6 to 7 : 8

[✓] Snap to Viewperiod — within: 9 minutes

[✓] Repeat Every 10 day(s) Mon Tue Wed Thu Fri Sat Sun

[✓] Use Gap-to-Track Ratio 2.5 :1 [ ] Total hours — hours

Maximum Gap: 12 hours 13 min

Minimum Gap: 14 hours 15 min

[✓] Allow MSPA

Notes: This is the launch phase of the VOY1 mission

Phase range: 2001 yr 102 day 14 : 12 : 13

to: 2005 yr 202 day 23 : 59 : 59

Fig. 3: The graphical requirements editor.

- conflict indicators where tracks are claiming the same resources at the same time
- a mouse-over table display of the attributes of any displayed track

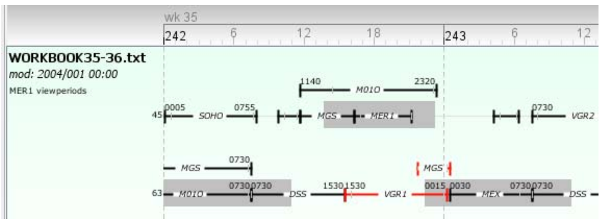
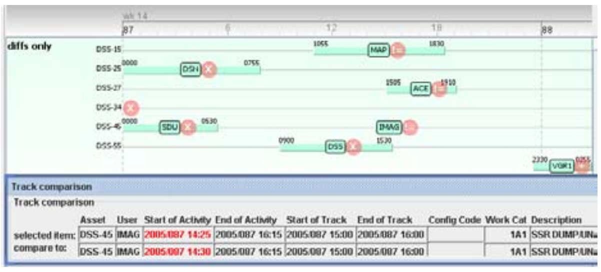


Fig. 4: Gantt view with annotations and conflict indicators

- a schedule difference display that visually highlights tracks added, tracks deleted, and tracks changed. Changed tracks can be clicked with the mouse to reveal a table display of the track attributes before and after, color encoded to show exactly which attributes have changed
- a metric-shaded background, to help visually spot regions of interest in the schedule. In the example shown, the shading is by gap duration in a 24h period, such that the bright shades of red indicate times with a scarcity of contacts.



Track comparison

Track comparison

	Asset	User	Start of Activity	End of Activity	Start of Track	End of Track	Config Code	Work Cat	Description
selected item:	DSS-45	IMAG	2005/087 14:25	2005/087 16:15	2005/087 15:00	2005/087 16:00		1A1	SSR DUMP/UN
compare to:	DSS-45	IMAG	2005/087 14:30	2005/087 16:15	2005/087 15:00	2005/087 16:00		1A1	SSR DUMP/UN

Fig. 5: Gantt view with schedule difference display

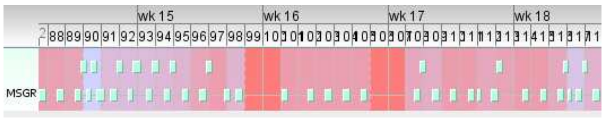


Fig. 6: Example of metric shaded background in the Gantt view

3.2 Local Heuristic Search

We have adapted the Aspen scheduler [10] to schedule all of the requirement types described above. This adaptation consists of two major components:

(1) a *mapping* from mission requirements (as described above) into Aspen's native modeling language [10]. A large portion of this mapping makes straightforward use of Aspen's built-in modeling features, including tasks and their hierarchical expansion into subtasks; temporal relationships; states; resources (consumable and nonconsumable); and parameters and their relationships. In some cases we have exploited Aspen's extension point mechanism to define external functions that can be invoked by Aspen. Details are provided in [11].

(2) a *set of heuristics* to drive Aspen's iterative repair-based search. These heuristics are based on "conflicts" in the schedule, such as state, resource, or temporal constraint violations, un-expanded hierarchical tasks, or un-propagated parameter dependencies. Part of the conflict selection and repair process incorporates randomness to help ensure a broad exploration of the search space. For a further description of the heuristics, see [11].

We have conducted experiments to gauge the performance of Aspen in this domain, with satisfactory results. In one experiment, we generated a full artificial schedule of 1861 tracks from periodic track requests in 39 CPU minutes. Modifications to this schedule were much faster: emergency antenna downtime or additional tracks were scheduled in 0.2 CPU seconds.

While the stochastic heuristic search technique embodied in Aspen has many advantages in this domain, it also has some disadvantages. The most important of these is that Aspen's search is not complete: it may fail to find a solution even if one exists. If solutions are sparse, the local search process may simply fail to find any in the time allotted (the "needle in a haystack" situation). To address this, we have complemented Aspen's search approach with several systematic search algorithms, as described in the next section.

3.3 Systematic Backtrack Search

We have implemented several systematic search algorithms based on constraint propagation and backtracking search. This approach, when appropriately limited in scope, can definitively answer the question of whether a solution exists and can additionally provide suggestions as to which requirements are overconstraining a problem, thus making it unsolvable.

The systematic search algorithms we have implemented are based on the following:

- search over time and resource assignments that start from the existing schedule (with or without constraint violations), so that alternatives "close" to the existing schedule are searched first
- tracks to assign are selected using a most constrained first heuristic
- backtracking is "fail fast", such that whenever any track has all domain values (times and resources) in conflict, the search immediately backtracks
- node and arc consistency is maintained based on track duration, view period windows, and resource availability

In addition to a basic backtracking search algorithm, we have also implemented an A* optimizing search, based on an objective that is a function of the time and resource assignments for each track. For the purposes of this discussion, the objective we consider is a "minimum change" measure, i.e. we seek the schedule with the smallest possible shifts in track times and antenna assignments.

Systematic search can be exponentially costly even with these techniques, so we bound its application by considering the following:

- limit to only a subset of the fully expressive requirements language
- limit the time horizon of the schedule considered
- limit the activities to consider, by locking a substrate of activities on the schedule and search only over a subset which is allowed to move
- limit the CPU time expended in the search

Our experience to date indicates that frequently there are "pockets" of conflicts that occur together, which have limited impact on the schedule as a whole. In this situation, systematic search can be bounded and yet still provide valuable insight into solution possibilities.

To investigate the performance of the systematic search algorithms, we ran a number of experiments on mid-range and short-term schedules. These first of these was on a 14 week period after the initial conflict resolution process. We considered each day in the schedule separately, and ran three different systematic search algorithms on each:

- **BT1**: standard depth-first backtrack search
- **BT2**: same, but tracks were prohibited from changing their antenna assignments
- **A***: search for all "minimum change" schedules

The results are illustrated in Fig. 7. Of the 98 days examined, 28 had track resource/timing conflicts, ranging from one to four per day. Of those 28 days, one had no feasible solution and would have to have requirements relaxed in some manner by the respective missions. The other 27 had solutions that were quickly found with BT1, in an average time of 10ms. The typical change was relatively small: an average 1.9±1.4 tracks were changed, with a total time shift (summed over all tracks) of 6.8±11.5 hours. BT2 only found one improvement over the solutions found by BT1. Most of the solutions found by BT1 (16 out of 27) were optimal by the "minimum change" metric described above. In 5 cases, the A* search found better solutions, and in 6 cases A* was stopped at a 1000s run time limit before finding any optimal solutions. The computational cost of A* was both higher and more varied than BT1 or BT2: the mean run time was 18±52s.

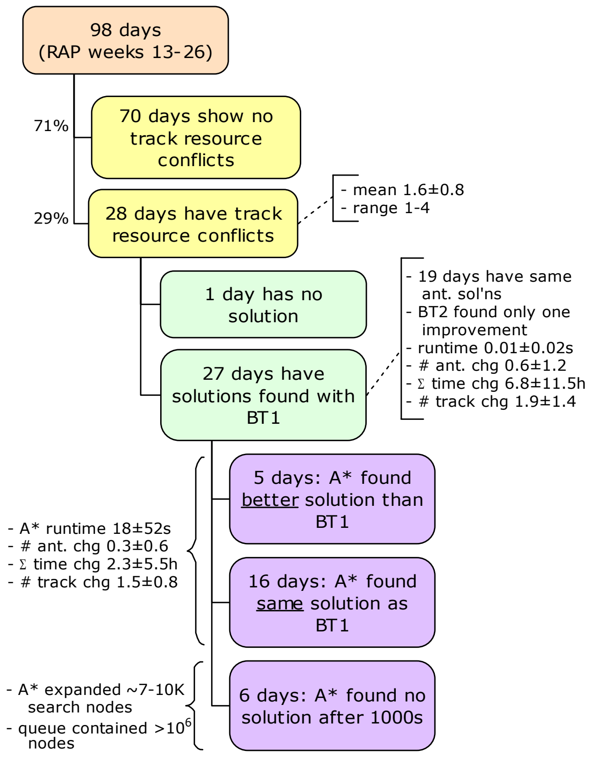

98 days (RAP weeks 13-26)
71%
70 days show no track resource conflicts
29%
28 days have track resource conflicts
- mean 1.6±0.8
- range 1-4
1 day has no solution
27 days have solutions found with BT1
- 19 days have same ant. sol'ns
- BT2 found only one improvement
- runtime 0.01±0.02s
- # ant. chg 0.6±1.2
- Σ time chg 6.8±11.5h
- # track chg 1.9±1.4
5 days: A* found better solution than BT1
16 days: A* found same solution as BT1
- A* runtime 18±52s
- # ant. chg 0.3±0.6
- Σ time chg 2.3±5.5h
- # track chg 1.5±0.8
6 days: A* found no solution after 1000s
- A* expanded ~7-10K search nodes
- queue contained >10^6 nodes

Fig. 7: Experimental results on a 98-day schedule span

As noted, this schedule represents the time after initial conflict resolution has been worked on by the schedulers and the mission users. We also selected a 2-week schedule snapshot taken before this conflict resolution process and ran the same experiments. The results are shown in Fig. 8. In this case 13 of the 14 days show conflicts, all of which were quickly resolved with BT1. The typical changes were much larger, however: an average 4.0±4.2 tracks were changed, with a total time shift of 19.7±26.9h. This reflects the smaller effort expended to resolve conflicts at this stage in the scheduling process. The A* algorithm found better solutions for 6 days, and no improvements for another 4 days. The A* runtime showed greater variability at 71±229s.

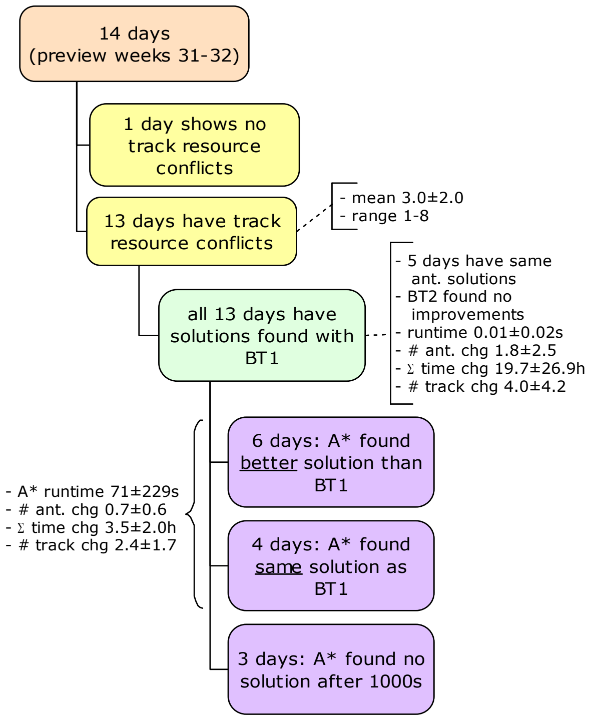

14 days
(preview weeks 31-32)
1 day shows no track resource conflicts
13 days have track resource conflicts
- mean 3.0±2.0
- range 1-8
all 13 days have solutions found with BT1
- 5 days have same ant. solutions
- BT2 found no improvements
- runtime 0.01±0.02s
- # ant. chg 1.8±2.5
- Σ time chg 19.7±26.9h
- # track chg 4.0±4.2
6 days: A* found better solution than BT1
4 days: A* found same solution as BT1
3 days: A* found no solution after 1000s
- A* runtime 71±229s
- # ant. chg 0.7±0.6
- Σ time chg 3.5±2.0h
- # track chg 2.4±1.7

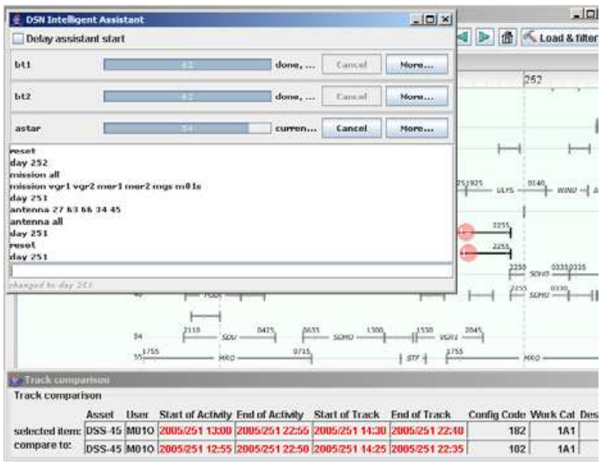
Fig. 8: Experimental results on a 14-day schedule before conflict resolution

3.4 Intelligent Assistants

The conclusion from these experiments is that systematic search can help identify solution possibilities when sufficiently bounded. We have integrated this functionality into the user interface elements described above by implementing intelligent assistants, consisting of:

- a *filtering mechanism*, currently text-based, to allow the user to focus attention on a subset of the schedule: by time, mission, resource, etc.
- multiple *background asynchronous search processes* that run on the filtered problem and utilize the systematic search algorithms described above.

A snapshot of this interface is shown in Fig. 9. The intelligent assistant background processes run automatically, starting whenever the filter criteria are changed by the user. They may be canceled by the user at any time, and they display current status as they run, both in a progress bar and in a textual status message. The BT1 and BT2 algorithms complete so quickly in our experiments (<10ms) as to be essentially instantaneous. The A* algorithm is much slower and more variable in duration: the status displayed in the progress bar shows how many of the total tracks have been successfully scheduled so far. In each case, when complete, the user can mouse over the status display to see a popup window with more detail, and can then click the "More" button to open a Gantt chart view of the schedule generated by the algorithm, along with highlighted differences from the original schedule.



```
reset
day 252
mission all
mission vgr1 vgr2 mer1 mer2 mgs m01s
day 251
antenna 27 63 66 34 45
antenna all
day 251
reset
day 251
```

Track comparison

	Asset	User	Start of Activity	End of Activity	Start of Track	End of Track	Config Code	Work Cat	Des
selected item:	DSS-45	M01O	2005/251 13:00	2005/251 22:55	2005/251 14:30	2005/251 22:40	182	1A1	
compare to:	DSS-45	M01O	2005/251 12:55	2005/251 22:50	2005/251 14:25	2005/251 22:35	182	1A1	

Fig. 9: Intelligent assistant dialog and display

By being unobtrusive, the intelligent assistants can provide the user with additional information at the precise time when the user is interested. Results are displayed immediately when they are calculated, and the user can cancel an ongoing search simply by revising the search criteria, which also automatically starts a new search using the new criteria. If the assistant generates an interesting schedule, the user can "drill down" into the details in a very straightforward manner.

4. DISCUSSION

We have developed and described a "hybrid" approach to automating DSN scheduling. An overall schedule for some time period can be generated based on the complex merged requirements from the different missions: for this phase, we make use of the ASPEN framework which implements an iterative repair-based search strategy. We have complemented this with a set of systematic search algorithms that work on an appropriately bounded region of the search space. We have embedded these algorithms in intelligent assistants that run unobtrusively to provide timely information to the user engaged in rescheduling and resolving schedule conflicts. The intelligent assistants can provide feedback immediately as they reach their conclusions. For example, it may be possible for an intelligent assistant to rapidly determine that there are no feasible solutions to a specific conflict situation, so a user can move on to consider changing the requirements or negotiating with another mission. Similarly, the assistants may identify an optimal set of choices to resolve a conflict so that the user's time can be better spent in choosing among them rather than trying to find a solution by manual experimentation. We have found that, while systematic search does not always find timely solutions, it frequently does very quickly provide valuable help to the interactive user.

Future research and implementation directions include the following:

- further integration of the iteratative repair and systematic search engines, to best exploit the advantages of each in a broader range of circumstances
- investigation of modelling and performance tradeoffs in mapping mission requirements into representational elements for the scheduler engines
- incorporating additional optimization criteria into the systematic search algorithm, to provide users with more insight into potential schedule modifications

---

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. Kan, E.J., J. Rosas, and Q. Vu, *Operations Mission Planner - 26M User Guide Modified 1.0*. 1996, JPL Technical Document D-10092.

2. Bell, C., *Scheduling Deep Space Network Data Transmissions: A Lagrangian Relaxation Approach*. 1992, JPL Technical Report.

3. Chien, S.A., et al. *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*. 1997. Albuquerque, NM.

4. Borden, C., Y. Yang, and G. Fox. *Planning and Scheduling User Services for NASA's Deep Space Network*. in *1997 International Conference on Planning and Scheduling for Space Exploration and Science*. 1997.

5. Werntz, D., S. Loyola, and S. Zendejas. *FASTER - A tool for DSN forecasting and scheduling*. in *Proceedings of 9th AIAA Computing in Aerospace Conference*. 1993. San Diego, CA.

6. Barbulescu, L., et al., *Scheduling Space-Ground Communications for the Air Force Satellite Control Network.* Journal of Scheduling, 2004. **7**(1): p. 7-34.

7. Barbulescu, L., L.D. Whitley, and A.E. Howe. *Leap Before You Look: An Effective Strategy in an Oversubscribed Scheduling Problem*. in *AAAI 2004*. 2004.

8. Frank, J., et al. *Planning and Scheduling for Fleets of Earth Observing Satellites*. in *I-SAIRAS 2001*. 2001.

9. Beck, J.C., et al. *Texture-Based Heuristics for Scheduling Revisited*. in *AAAI 1997*. 1997.

10. Chien, S., et al., *ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling*, in *SpaceOps 2000*. 2000: Toulouse, France.

11. Clement, B.J. and M.D. Johnston. *The Deep Space Network Scheduling Problem*. in *IAAI*. 2005. Pittsburgh, PA: AAAI Press.