The Deep Space Network Scheduling Problem

Bradley J. Clement and Mark D. Johnston

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
bclement@jpl.nasa.gov, mdj@jpl.nasa.gov

Abstract

We describe the Deep Space Network's scheduling problem based on a user requirement language. The problem is difficult to encode by almost all existing planning and scheduling systems. We describe how it can be mapped into a system that supports metric resources, durative action, simple temporal network constraints, and task hierarchy among other language features. We also describe how we adapted a local search scheduler to generate schedules. However, we argue that the application will best serve the users if local search is combined with systematic search. We describe how an implemented systematic search can be effectively applied to rescheduling.

Introduction

The Deep Space Network (DSN) maintains 16 antennas (26 to 70m in diameter, Figure 1) that provide tracking, navigation, and data transmission services among others. Antenna complexes are located in Goldstone, CA, Madrid, and Canberra and provide services to spacecraft within and beyond the gravitational influence of Earth. While 150 missions are listed as DSN users, about 20 spacecraft are allocated resources in a 4-month schedule.

Schedules are currently manually generated a year into the future with allocations to the minute. These are currently generated a week at a time and average 370 *tracks* (allocation of an antenna to a mission over a time period) per week. These tracks are typically 1 to 8 hours long and must be allocated in a *viewperiod* (i.e. a time period when the spacecraft is visible to the antenna). There are around 1650 of these viewperiods defined per week. The DSN's goal is extend the schedule out to ten years where they are currently just predicting and adjusting resource loading based on coarse requirements. The near-term schedule (within 8 weeks) additionally considers the allocation of service-specific equipment, personnel controlling the antennas, and some additional geometric constraints on pointing the antenna. In this paper we describe how we are working with missions to employ scheduling algorithms to generate and repair mid-range (8 weeks to 6 months) and long-term (beyond 6 months) schedules.

Figure 1. Deep Space Network facility

While the schedule is fairly large (~2000 tracks and ~8000 viewperiods in a year), the greatest inefficiency of the current scheduling process is not in generating low-conflict schedules but in trying to meet the ill-specified requirements of the missions. The requirements used for generating schedules are only specified in hours/week per antenna. DSN schedulers (personnel) can only guess the acceptable duration, frequency, and alternative antennas of the tracks generated. While missions convey some of this information over the phone or e-mail to schedulers, there are no records of this additional requirement information, and subsequent proposed schedule changes often violate the intended requirements. These changes often affect multiple missions, resulting in frequent meetings, e-mails, and phone calls to rework the schedule. At the time of this

Work 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.

paper, there are six members of the mid-range scheduling staff and four near-term schedulers who work with 13 mission scheduling representatives to the DSN.

In general, space applications often differ from others in that scheduling requires rich languages for modeling temporal relationships, spacecraft instruments, and dynamics (e.g. Eggemeyer *et al.* 1990, Muscettola *et al.* 1998, Chien *et al.* 2004, and Ai-Chang *et al.* 2004). For the long- and mid-range DSN scheduling, there are no complex resources to model. However, mission requirements for DSN resources are not trivial to model as this paper explains.

There has been much work aimed at automating the DSN scheduling process. For many years, the Operation Mission Planner (OMP-26) used heuristic search to allocate 26-meter antennas to missions and linear programming to adjust track time intervals (Kan *et al.* 1996). Other automated scheduling tools were research projects and were never deployed. LR-26 is a customizable heuristic scheduling system also for the 26-meter antennas using Lagrangian relaxation and constraint satisfaction search techniques (Bell 1997). The Demand Access Network Scheduler (DANS) expanded the scope to include all antennas using a heuristic iterative repair approach (Chien *et al.* 1997). These systems schedule mission requirements of the form, "four 15-minute tracks every day." In this paper, we consider an approach that combines some of the strengths of these systems. Other GUI planning tools that have been used for forecasting, analysis, and manual scheduling include TIGRAS (Borden *et al.* 1997) and FASTER (Werntz *et al.* 1993).

Other systems have investigated oversubscribed scheduling problems that capture the basic constraints of DSN's mid- to long-term resource allocation. The Air Force Satellite Control Network (AFSCN) also schedules satellite communications requests, on a larger number of satellites and ground stations, but limited to one day at a time. Requirements are more simply specified as an ordered list of resource and time window pairs (Barbulescu, Watson *et al.* 2004), (Barbulescu, Whitley *et al.* 2004). For this problem, which exhibits "plateaus" in the search to minimize the number of conflicting activities, local repair techniques have been found less effective than approaches which make more moves at once. Another satellite scheduling problem is that of fleets of Earth observing satellites where the activities to schedule have similar kinds of viewperiod constraints, but additionally require onboard resources such as instruments and data recorders (Frank, Jonsson *et al.* 2001). In this problem, however, the different requests are prioritized, and the goal of finding a "best" subset to fit on the schedule can be addressed with a greedy approach using texture-based heuristics, as in e.g. (Beck, Davenport *et al.* 1997).

In the DSN scheduling problem, requirements from users are much more complex. Unlike simple window constraints, the DSN requirements are frequently non-local in that they specify conditions not on single communication contacts but on the global properties of an entire sequence of contacts (such as average gap to track ratio and minimum and maximum gap sizes). The number of activities and the duration are dependent parts of the search space. In addition, the requirements implicitly specify tradeoffs in the form of AND/OR trees of requirements. This makes for a much richer capability for users to specify their real requirements and alternatives, while providing the scheduling system with additional options for generating and optimizing the schedule.

These kinds of requirements are also augmented with preferences on timing, duration, number of activities, time between activities, etc. In addition, because of the human negotiation overhead, a basic metric in resolving scheduling problems is to limit the number of parties affected to simplify negotiation.

Other scheduling applications also share this need to capture and optimize the objectives of multiple users. The SPIKE system is used to schedule Hubble Space Telescope observations for many scientists (Johnston and Miller 1994). When the scientists' requirements are not clearly elicited, the resulting schedule may be inadequate. Although SPIKE provides an extensive vocabulary for constraints, it is limited in expressing scheduling goals, alternatives and tradeoffs. As a result, scientists must be re-involved to adjust their requirements for successive re-planning phases. In fact, clearly capturing the goals of scientists competing for spacecraft resources is an aspect of any space mission. Military missions also share the difficulty of defining how goals should be met.

Market-based approaches have been proposed to address some aspects of this problem, in that tradeoffs will be managed by competition for resources by users who best understand their own needs, and are best qualified to make tradeoffs (Wellman, Walsh *et al.* 2001, Etherton, Steele *et al.* 2004). However, these approaches present the difficulty of defining and allocating a common "currency", and

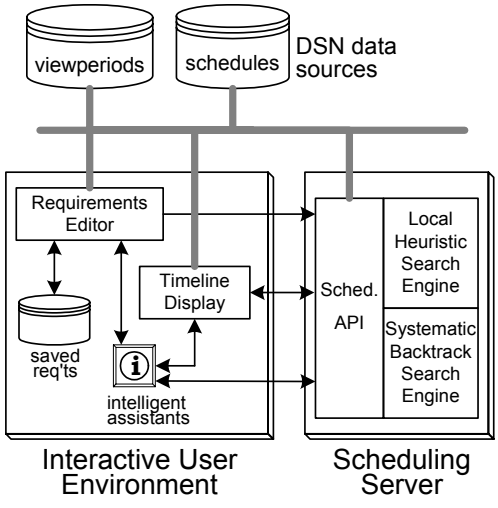

Figure 2. System overview

ensuring that optimization by individual users also optimize the system as a whole.

What is lacking in these application domains is an expressive goal specification language and a planner/scheduler tailored to handle them. In this paper, we describe a requirement language for the customers of the DSN, an adaptation of a local search scheduler that can schedule the requirements, and a backtracking search algorithm for resolving isolated conflicts with minimal perturbation to the plan. The basic strategy is to use local search in conjunction with systematic search to exploit their complementary strengths. Our implementation focuses on meeting hard constraints instead of optimization but we also describe optimization criteria as a research challenge. A diagram of our prototype system (Figure 2) shows a user environment where requirements are elicited and translated into a scheduling problem. The translation is a major focus of this paper. The goal is to have the scheduling algorithms work together to provide alternative Pareto-optimal solutions based on requirements and preferences to be suggested in a non-obtrusive manner.

DSN Scheduling Problem

First we describe the user-independent constraints on the DSN scheduling problem. Then we describe the requirement language to capture mission-specific constraints and mention user preferences as criteria for optimization. Lastly, we discuss the ability of current planning and scheduling systems to represent these complex goals.

Constraints

The basic constraints of DSN scheduling are that no two spacecraft can use the same antenna at the same time, and a spacecraft can only use an antenna if it is within view. An exception is that two spacecraft can use the same antenna if they are not both uplinking, and the antenna can point to them both, i.e. they are located in the same direction from Earth. So, a Mars spacecraft can share an antenna with another Mars spacecraft. This sharing can be modeled by making antennas non-consumable resources[1] (e.g. machines and power) with a capacity of five, and sharing tracks just use three units of the antenna if uplinking and two units if not, while non-sharing tracks use five units.

There are additional constraints on the timing of tracks. Each track has setup and teardown durations, during which equipment is (dis)connected and antennas slew to the target. The setup, track, and teardown together are called an *activity*. At the Goldstone complex, no track can start at when there are two or more other tracks starting within the interval 15 minutes before and 15 minutes after the track's start. The same rule applies to the start of activities. There are two exceptions. When two activities share an antenna and start at the same time, their start time is counted as one track instead of two. Also, the Cluster spacecraft (a constellation of four spacecraft) do not cause conflicts with each other. At the other complexes, the only timing restriction is that no two activities can start within five minutes of each other.

These timing constraints can be modeled with a non-consumable resource with capacity of three for Goldstone and one for Madrid and Canberra, and one unit of the resource is used during the 30 (Goldstone) or 10 (Madrid & Canberra) minute interval centered on the start time of each track and/or activity.

Requirements

We now describe mission-specific requirements for DSN resources and mention some possible ways to represent them in a planner/scheduler.

Requirements are connected in an AND/OR tree in order to provide DSN customers the ability to represent alternative choices and grouping. So, an AND requirement means that all of the sub-requirements must be met, and an OR requirement means that only one sub-requirement must be met. Hierarchical planners allow tree structures to be modeled explicitly, e.g. UMCP (Erol, Nau *et al.* 1994), SHOP (Nau, Au *et al.* 2003), ASPEN (Chien, Rabideau *et al.* 2000), EUROPA (Frank, Jonsson *et al.* 2001).

Missions must be able to specify alternative antennas for a track or the use of multiple antennas for arraying, tracking, etc. The tree hierarchy can be used to capture these constraints. Missions must also be able to give ranges of acceptable start times and durations for a track. The planner/schedule must be able to represent and enforce durative actions and simple temporal network constraints (Dechter, Meiri *et al.* 1991). Most missions also require that start times and durations of tracks and activities are divisible by five minutes. One way to handle this is to have a non-consumable resource that has a very large capacity to accommodate any number of tracks that may use a unit at the same start time and is initialized to be empty at all times except at the time units divisible by five minutes. In addition, a mission must be able to lock in on an allocated time or antenna to prevent schedule changes. This could be done simply by modifying the AND/OR hierarchy to only include the allocated antenna(s), and shrink any time ranges to the scheduled value. However, this could be inconvenient if values are then unlocked to allow to reallocation. Some planners (e.g. ASPEN (Chien, Rabideau *et al.* 2000)) have permissions on how actions can be added, modified, or removed. For example, locking and unlocking the timing of a track is easily done by removing or adding a *move* permission.

There are also many flavors of repeated track requirements that can be generalized as a specification of an initial start time, end of overall period, a number of tracks, total duration of tracks, duration of individual tracks, and the time gap (or overlap) of neighboring tracks. Ranges of acceptable values can be specified for each of

[1] These are resources whose usage amount is restored at the end of use as opposed to consumable resources like fuel where the reduced level persists after use.

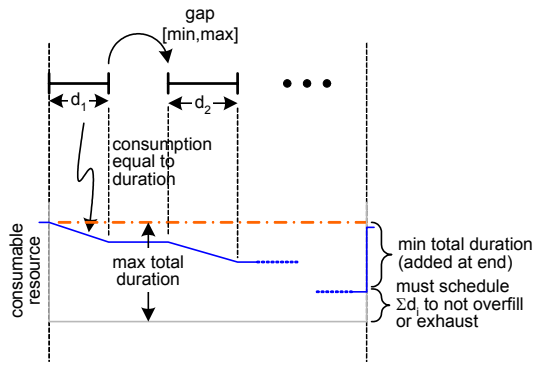

Figure 3. Modeling a repeated track

these variables. As mentioned before the ability to enforce STN constraints is required for ranged time values and gaps. Enforcing a total duration and number of tracks can be done by creating a consumable resource tracking total duration for each periodic requirement with a capacity of the maximum total duration. At the initial start time, the resource is full, and each track uses the resource an amount equal to its duration. At the end of the overall period, the minimum total duration is restored to the resource. Thus, the planner must add in tracks to meet the minimum total duration, or the resource will fill beyond capacity at the period end. This model is illustrated in Figure 3. Another way to model this repeated track constraint is to create a recursive AND/OR hierarchy, each level either creating a track or ending recursion. The recursion may end when the total duration is in satisfactory range. Of course, the planner/scheduler must have a way of tracking total duration (possibly through a duration resource as just described) and using it to control the OR branch to end recursion.

Another requirement type specifies that one set of tracks overrides another set. For example, a probe may have a requirement to periodically downlink telemetry every 4 hours, but during a navigation phase, there will be a 24-hour continuous track that interrupts the downlinks. This means that when an overriding track overlaps (in time) an overridden track, the overridden track is canceled, but the requirement for it remains satisfied. This can be modeled by creating an override non-consumable resource for each track that has a very large capacity to accommodate consumption by a maximum number of overriding tasks. If the resource is not full (meaning the track is overridden), then the track action does not use antenna resources (possibly through controlling an OR branch). This assumes that the planner/scheduler can use resource values to determine resource consumption.

Preferences

Missions could have preferences over alternative OR branches, antenna choices, or timing and durations in all of the previously described requirements. They could also have preferences over combinations of these values. Eliciting these kinds of preferences is a research sub-field of its own and outside the scope of this paper. However, in making changes to an existing schedule, missions have additionally expressed interest in minimizing the number of changes to the schedule, the number of affected missions, and (maybe most importantly) the number of people that will be involved in the negotiation (since some missions share DSN scheduling representatives).

Modeling Languages

Based on the constraints and requirements for this problem, a planning or scheduling system would need to be able to encode non-consumable metric resources, STN constraints, durative actions, AND/OR action hierarchy, conditional OR branching, exogenous events (scheduled actions that cannot be modified, e.g. viewperiods), and potentially arbitrary code hooks into action parameters. The only systems that represent these constructs explicitly are ASPEN (Chien, Rabideau *et al.* 2000) and EUROPA (Frank, Jonsson *et al.* 2001). IxTeT (Laborie and Ghallab 1995) only lacks OR branching, and SHOP (Nau, Au *et al.* 2003) only lacks durative action. This does not mean that the DSN problem cannot be encoded by these planners or that they cannot be effectively used, but many times the lack of such constructs can result in a large expansion of the encoding that overcomes system limits.

Schedule Generation

We tailored the ASPEN planning system (Chien, Rabideau *et al.* 2000) to schedule all of the requirements mentioned earlier. This is done by modeling states, resources, and hierarchical tasks; translating existing DSN schedule and viewperiods into ASPEN problem instances, and choosing and adapting heuristics to guide search. Before describing the adaptation, we give some background on ASPEN's scheduling algorithm.

Iterative Repair in ASPEN

ASPEN is a local search, heuristic iterative repair planner/scheduler—it takes an initial schedule and iteratively adds, removes, reschedules, and refines tasks in the schedule for as long as it is invoked, until all conflicts are resolved, or until a perfect utility is obtained. It is early commitment in that it grounds the timing and effects of tasks as early as possible. No backtracking states are kept, so search states may be repeated, but the required memory grows only with the size of the schedule.

ASPEN's modeling language is well-suited to capture DSN requirements. In addition to the constructs described in the Modeling Languages section, parameters in tasks can be computed as functions of other parameters in the task or in related tasks. Many basic functions are built in, but modelers can also write C++ functions.

Conflicts in ASPEN can be state, resource, or temporal constraint violations, un-detailed hierarchical tasks, or un-propagated parameter dependencies (among other things).

Preferences can be based on many statistics on almost any variable in the schedule (e.g. task, parameter, resource value). In one iteration of repair, ASPEN selects a conflict or preference, selects a repair or improvement method (e.g. add, move, delete, detail), chooses a task to which to apply the method, and chooses how to apply it (e.g. where to add/move). These iterations are repeated until the aforementioned end criteria.

Each of these selections and choices in an iteration is guided by a heuristic and is often made stochastically. Heuristics are also selected based on probabilities defined by the modeler for the domain.

Modeling Requirements

Requirements are defined in ASPEN's modeling language either as an abstract requirement in the AND/OR tree, a simple track requirement, a periodic track requirement, a segmented track requirement, or an override requirement. If not stated otherwise, constraints and requirements are modeled as suggested in the Constraints and Requirements sections.

A **simple track requirement** is an abstract task that contains the track within a window where the start time and duration of the track can range as specified and enforced by temporal constraints. The child of this task is movable and decomposes into an AND/OR tree for choices of single or multiple antennas. The leaves of this tree use the antenna resource according to the antenna sharing rules mentioned in the Requirements section and require that the track be within a viewperiod state for the spacecraft and antenna.

A **periodic track requirement** is a popular simple case of the repeated track requirement described generally in the Requirements section. Simple track requests are repeated within a time window with a fixed period between start times. Recursive decomposition is used to generate a specified number of tracks within the window as described in the Requirements section. Duration resources were not used because enforcing temporal constraints between activities that are being added and deleted is difficult to engineer. The number of tracks and window are made consistent by the user-interface.

A **segmented track requirement** has all of the complexities of the repeated track requirement described in the Requirements section. It is differentiated from the periodic track because it was more convenient here to use decomposition constraints in place of simple track requirements to regulate to "wiggle room" for individual tracks. Also, the interaction of the number of tracks and track durations in the search space requires different heuristics. The user interface ensures consistent input parameter values. Parameter dependencies passed through decompositions track the summed duration of tracks to determine whether to stop recursion.

An **override requirement** is a task that overrides another. In order to override a set of tracks, the parent requirement can be overridden. If there is no common parent, then separate override requirements must be specified. At the leaves of the requirements an OR branch is inserted to only use the antenna if not overridden. Using resources to detect when tracks are overridden (as described in the Requirements section), can slow scheduling operations that check for violated resources. Thus, we instead created a static table in user-defined parameter dependency code (C++) to track the time periods that requirements are overridden. The OR branch decomposition choice is tied to a parameter dependency function that checks the table to see if it is overridden. This dependency function is triggered whenever the override requirement or the potentially overridden track is added, moved, or resized in duration.

Heuristics

Most heuristics listed below were written specifically for this problem in C++ but build on built-in heuristics. Some are too involved to describe in detail.

- half of the time choose the conflict to repair based on a default measure of badness for the conflict type; otherwise choose the earliest occurring
- for violated constraints choose the move repair method 70%; abstract and detail 30%
- for antenna conflicts, move 53%; choose antenna 47%
- for viewperiod conflicts, move 53%; choose antenna 47%
- choose antennas within view ten times more often than those not in view
- move (probabilistically) near to current time when not causing conflicts and resolving current conflicts (preferring viewperiod coverage to available antenna intervals), but move to a random legal time 1% of the time
- durations are maximized within legal viewperiod and available antenna intervals 60%; just within viewperiod 30%; a default specified value 3%; chosen randomly 3%; maximum allowed 2%; minimum allowed 2%

For segmented track requirements, the durations are deterministically chosen maximized within legal viewperiod and available antenna intervals.

Preliminary Performance Results

To gauge performance of scheduling requirements, we generated an artificial schedule of 1861 tracks from periodic track requests in 39 CPU minutes (resolving 2305 initial view period/antenna conflicts). It reschedules to accommodate individual emergency tracks in 0.2 CPU seconds and emergency antenna downtime in 0.2 seconds. It handles doubling of one mission's track requests over one week (to 42 total) in 2.7 seconds. Scheduling segmented track requirements is an order of magnitude slower. Our future work will compare different modeling approaches for the segmented track requirements to find a more efficient alternative.

Rescheduling

While the heuristic local search technique embodied by ASPEN has many advantages when applied to this domain, it also has some drawbacks. The most significant of these is that failure to find a solution does not mean that no solution 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 complement the ASPEN local search technique, we have also implemented a systematic search algorithm employing constraint propagation and backtracking. This approach, when limited in scope as described below, provides some uniquely useful information to users of the system.

- it can definitively answer the question of whether any solution exists that satisfies the requirements and constraints
- it can provide suggestions as to which requirements (or their combinations) are overconstraining a problem, thus making it unsolvable

This latter information can be used to guide the heuristic search in ASPEN, and we are planning in the future to incorporate this information in an automated way.

The systematic search algorithm we have implemented and tested is based on the following:

- constraint propagation — node and arc consistency is maintained based on track duration, view period windows, and resource availability
- search over time and resources starts from the existing schedule (with or without constraint violations), so that alternatives that are "close" to the existing schedule are searched first
- most constrained first activity selection
- fail fast backtracking — whenever any unscheduled activity has all domain values in conflict, the search immediately backtracks

Even using these techniques, systematic search can be exponentially costly in time/space and so we apply it only in a bounded manner to a specific portion of the problem. These bounds are established in up to four ways: consider only a subset of the fully expressive requirements language (such as limiting the depth of the and-or tree); horizon-limited (e.g. to a single day of the schedule time span); activity-limited (locking a substrate of activities in place on the schedule and searching over a subset that is free to move), and effort-limited (bounding the CPU time expended if no solution is found). Our experience to date indicates that conflicts often occur in "pockets" of activities (e.g. all contending for the same antenna at the same time) that have localized effect on the schedule as a whole, and that systematic search can quickly find adjustments that resolve the problem, or determine that no such adjustments exist. We expect that the systematic search algorithm will be particularly useful in the context of intelligent assistants that help users understand the nature of unsatisfied requirements or constraint violations, and then come up with alternatives to be negotiated with other DSN users.

This search algorithm was experimentally applied to the DSN mid-range schedule for the second quarter of 2005. Only about one out of four days had an average of 1.4 conflicts, and the algorithm found solutions in a small fraction of a second, usually requiring between one and five schedule changes. However, this schedule has already been mostly de-conflicted by the scheduling personnel. We also applied the algorithm to a two-week schedule freshly generated before de-confliction. All but one day has an average of 2.0 conflicts, and there are usually between one and eight schedule changes in the solution. The algorithm still ran in a small fraction of a second.

We have also run experiments on a two-week "Workbook" schedule, representing a time period which is in active conflict resolution by the missions and the DSN scheduler staff. We found an average number of 2.4 tracks in conflicts per day over this period, with 3 days having no conflicts. One day had no solution, but the remainder had solutions that were quickly found (in small fraction of a second) by the systematic search algorithm. While most days were solvable by shifting 1 or 2 tracks, there two days that required 6 or more track changes to resolve all the conflicts. These latter cases could be difficult for human users to solve without automated help.

A planned extension of the systematic search algorithm will incorporate optimization over the various objectives that are important to DSN users. This is an advantage that systematic search provides over local heuristic search, which may become trapped in a local optimum. For example, in DSN scheduling we may need to find feasible solutions that:

- minimize changes to the current schedule
- minimize the number of missions impacted by proposed schedule changes
- schedule tracks as close as possible to the middle of their viewperiods
- maximize setup activities that occur during shifts when expert personnel are available

In addition, individual missions will have their own specialized optimization objectives. Systematic search offers the potential to optimize (over a bounded problem) and thus, for example, to answer the question of whether any improvement to a particular schedule is possible. This provides valuable feedback to a DSN user, who can then direct attention to other issues rather than fruitlessly seek additional improvements.

Conclusion

We described the DSN scheduling problem in terms of constraints, requirements, and preferences that are difficult to encode in most (if not all) planning and scheduling systems. We described one way of mapping the constraints and requirements into a local search planner/scheduler and mentioned some alternatives. One simplifying aspect of the problem is that the absence of consumable resources can enable a scheduler to focus on a narrow time frame to reschedule activities. We assert that

a systematic scheduling algorithm capable of handling a subset of the identified modeling constructs would have a distinct advantage in rescheduling for more focused problems over the more common non-systematic approaches applied to oversubscribed scheduling problems that have been largely used in the past for rescheduling. Our implementations of the local search and systematic planners show promise with reasonable performance.

References

Ai-Chang, M., J. Bresina, L. Charest, A. Chase, J. C.-J. Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. G. Chafin, W. C. Dias and P. F. Maldague (2004). "MAPGEN: mixed-initiative planning and scheduling for the Mars Exploration Rover mission." IEEE Intelligent Systems 19(1): 8-12.

Barbulescu, L., J.-P. Watson, L. D. Whitley and A. E. Howe (2004). "Scheduling Space-Ground Communications for the Air Force Satellite Control Network." *Journal of Scheduling* **7**(1): 7-34.

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

Beck, J. C., A. J. Davenport, E. M. Sitarski and M. S. Fox (1997). "Texture-Based Heuristics for Scheduling Revisited". AAAI 1997.

Bell, C. (1992). "Scheduling Deep Space Network Data Transmissions: A Lagrangian Relaxation Approach," Technical Report, Jet Propulsion Laboratory.

Borden, C., Y. Wang, and G. Fox (1997). "Planning and Scheduling User Services for NASA's Deep Space Network," Working Notes of the 1997 International Workshop on Planning and Scheduling for Space Exploration and Science.

Chien, S., R. Lam, Q. Vu (1997). "Resource Scheduling for a Network of Communications Antennas," IEEE Aerospace Conference. Aspen, CO.

Chien, S., G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins and D. Tran (2000). ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling. *SpaceOps 2000*. Toulouse, France.

Chien, S., R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, R. Lee, D. Mandl, S. Frye, B. Trout, J. Hengemihle, J. D'Agostino, S. Shulman, S. Ungar, T. Brakke, D. Boyer, J. VanGaasbeck, R. Greeley, T. Doggett, V. Baker, J. Dohm, F. Ip (2004). "The EO-1 Autonomous Science Agent," Proceedings of AAMAS.

Dechter, R., I. Meiri and J. Pearl (1991). "Temporal Constraint Networks." *Artificial Intelligence* 49: 61-96.

Eggemeyer, C., S. Grenander, S. Peters and A. Amador (1997). Long Term Evolution of a Planning and Scheduling Capability for Real Planetary Applications. Working Notes of the First International Workshop on Planning and Scheduling for Space, Oxnard, CA.

Erol, K., J. Hendler and D. Nau, *Semantics for hierarchical task-network planning*, 1994, University of Maryland Technical Report CS-TR-3239.

Erol, K., D. Nau and J. Hendler (1994). "UMCP: A Sound and Complete Planning Procedure for Hierarchical Task-Network Planning". AIPS 1994.

Etherton, J., I. A. Steele and C. J. Mottram (2004). "A free market in telescope time?" in *Optimizing Scientific Return for Astronomy through Information Technologie (SPIE Vol. 5493)*. P. J. Quinn and A. Bridger, SPIE. 5493**:** 90-96.

Frank, J., A. Jonsson, R. Morris and D. E. Smith (2001). "Planning and Scheduling for Fleets of Earth Observing Satellites". I-SAIRAS 2001.

Gerevini, A., A. Saetti and I. Serina (2003). "Planning through Stochastic Local Search and Temporal Action Graphs." *Journal of AI Research* 20: 239-290.

Johnston, M. D. and G. E. Miller (1994). Spike: Intelligent Scheduling of Hubble Space Telescope Observations. In *Intelligent Scheduling*. ed. M. Zweben and M. Fox. San Mateo, Morgan Kaufmann**:** 391-422.

Kan, E., J. Rosas, and Q. Vu (1996). "Operations Mission Planner - 26M Users Guide Modified 1.0", JPL Technical Document D-10092, April.

Laborie, P. and M. Ghallab (1995). "*Planning with Sharable Resource Constraint*". in IJCAI 1995.

Muscettola, N., P. Nayak, B. Pell, and B. Williams (1998). "Remote Agent: To Boldly Go Where No AI System Has Gone Before," *Artificial Intelligence* 103(1-2):5-48, August.

Nau, D., T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu and F. Yaman (2003). "SHOP2: An HTN Planning System." *Journal of AI Research* 20: 379-404.

Vidal, V. and H. Geffner (2004). "Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming". in AAAI 2004, San Jose, California.

Wellman, M. P., W. E. Walsh, P. R. Wurman and J. K. MacKie-Mason (2001). "Auction Protocols for Decentralized Scheduling." *Games and Economic Behavior* 35: 271-303.

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