Conflict- and Fairness-Directed Heuristic Search Scheduling for NASA's Oversubscribed Deep Space Network

Mark D. Johnston
Jet Propulsion Laboratory – California Institute of Technology
mark.d.johnston @ jpl.nasa.gov

Abstract

This paper describes a comparison of search methods applied to the same problem of scheduling activities for NASA's Deep Space Network (DSN). The DSN consists of large (34- and 70-meter) radio antennas at three sites around the world, and provides communications and navigation support for dozens of space missions. The DSN is oversubscribed – more time is requested than can be accommodated – and it is an ongoing challenge to schedule the available resources while also balancing the satisfaction of requests from the disparate set of users. As part of a study of state-of-the-art solution techniques for oversubscribed scheduling, a standard set of problems was defined, and solutions were attempted with several classes of algorithms developed by different teams. These included variants of Quantum Annealing (QA), Mixed-Integer Linear Programming (MILP), and Constraint Satisfaction Problem (CSP) based heuristic search (CSP-HS). The results of this comparison, first reported in this work, show that the heuristic search method CSP-HS significantly outperforms the other methods in the study in terms of solution quality and runtime performance, when objectives are to maximize the "fairness" of the resulting schedule (e.g. minimize starving some users at the expense of others) while simultaneously clearing all constraint violations and fitting as much as possible into the schedule. CSP-HS shows progressively better performance on more oversubscribed problems. The set of benchmark problems is made openly available for other groups to try.

1 Introduction

NASA's Deep Space Network (DSN) [1] consists of a set of large (34-meter and 70-meter) antennas located at three sites that are roughly equally spaced around the world: Goldstone, California USA; Madrid, Spain; and Canberra, Australia (Figure 1; see e.g. [Imbriale, 2003]). The DSN provides the sole communications lifeline for dozens of interplanetary spacecraft. It also provides navigation services and scientific observations in the fields of radio astronomy and planetary radar. It is the largest and most sensitive telecommunications network in the world. At the present time, the DSN supports about 40 users with 14 antennas, with the number of users roughly expected to double in the next 5 years. The DSN is operated for NASA by the Jet Propulsion Laboratory in Pasadena, California.

An an international asset in high demand, the scheduling of the DSN is heavily subscribed. The scheduling process for the DSN [Johnston *et al.*, 2014] is conducted on a rolling weekly basis, months ahead of execution. The first step in that process is to integrate and deconflict the many requests for services that users submit ahead of a common deadline. That process is one of the most labor-intensive and time-consuming aspects of DSN scheduling, and is the focus of the study reported here. It generally takes days of work to manually deconflict the schedule to a level where teams of schedulers spend another week or so negotiating final changes to the point where a mutually-concurred baseline schedule can be published. Missions generally require the schedule to be well-defined and stable weeks to months ahead of time to enable their own processes for planning and onboard sequencing of activities. These need to be consistent with opportu-

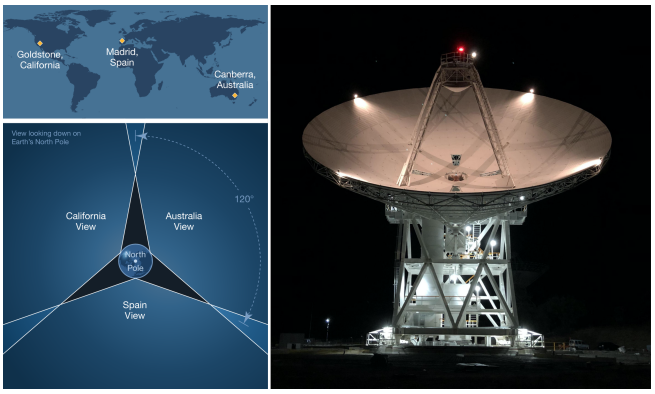

Figure 1: The Deep Space Network station locations (upper left), and approximate fields of view showing how a spacecraft is nearly always visible from at least one station (lower left); on the right is one of the DSN 34-meter antennas at Madrid.

nities to download data and to receive commands from the ground, often with long light travel time delays.

As part of an assessment of technologies that could potentially be used to improve this part of the process, a study was initiated in 2018 to assess new approaches to the solution of the same oversubscribed DSN scheduling problem. These included:

- Quantum Annealing (QA) inspired quadratic unconstrained binary optimization (QUBO) formulation [Guillaume *et al.*, 2022], solved with the D-Wave Leap Hybrid Solver Service (HSS)
- Δ-MILP [Sabol *et al.*, 2021; Claudet *et al.*, 2022], a Mixed-Integer Linear Programming variant
- CSP-HS, a Constraint Satisfaction Problem (CSP) based heuristic search technique, CSP-HS [Shouraboura *et al.*, 2016; Johnston, 2019; Johnston, 2020] that was previously under development for DSN scheduling for an investigation of priorities and user preferences, and was adapted to ingest the common problem format, and to implement the same scheduling objectives as used in the other methods.

In addition, a Reinforcement Learning (RL) [Goh *et al.*, 2021; Goh *et al.*, 2022] approach was also explored, but unfortunately did not yield final results during the time period of the study.

In the following we first describe the scheduling problem in more detail, including the relevant constraints and objectives that play a key role. We then describe the CSP-HS heuristic search algorithm. This is followed by a detailed comparison of the results from three of the methods that have solved at least a subset of the problems. The complete source of the problems from 2018 are publicly available [Goh *et al.*, 2022]. We conclude with a summary of current status and ongoing and future work.

2 DSN Scheduling

Scheduling a week of DSN activity requires the integration and deconfliction of requests from all DSN users. Requests are formulated in terms of requested tracking time, along with constraints such as temporal limits, specific sets of antennas that can support the requested services, min and max separation limits, etc.: for a full list see [Johnston *et al.*, 2014]. In addition to request-specific factors, scheduling depends on the line-of-sight visibility of the spacecraft by the antenna, and on whether the antenna is available (i.e. not down for maintenance or other upgrades). Other constraints determine whether missions that are simultaneously in view can actually be scheduled in parallel. While it is necessary to consider all of these factors in constructing and checking an operational schedule, they overly complicate an algorithm test scenario. As a result, a characteristic but somewhat simplified problem set was constructed based on requirements submitted by users as part of their long-range loading forecast.

2.1 Scheduling Problem Characteristics

The major features of the requests in this set are:

- a valid time range interval during which any generated tracks must be contained
- a tracking time duration
- a set of antennas that can accommodate the necessary services
- for longer tracks, a range of valid durations, and the option to split a track into segments with a specified minimum duration
- service-dependent setup time and teardown time that has to be included on all tracks (and on any split segments)
- for certain tracks, the option to schedule on either a single 70-meter antenna, or on a simultaneous array of two (or more) 34-meter antennas
- for navigation purposes, some tracks must be scheduled on two different antennas at once, and at different complexes

In addition to request specifics, there are general constraints based on the spacecraft involved:

- tracking time must be scheduled when spacecraft are in view of a particular antenna, and when the antenna is available (not in maintenance or other downtime); setup and teardown time can be scheduled out of view
- scheduled activities on one antenna (including setup and teardown) must not overlap with tracking the same spacecraft on another antenna (except for arrays or special navigation tracking)
- different spacecraft cannot generally be simultaneously scheduled in overlap on the same antenna – such a configuration is called a *facility conflict*

A typical week includes hundreds of requests covering a 7-day interval that must be fit into the week without violating any of the constraints listed above. Such a schedule is called *feasible*. A total of 6 sample problems were defined for the investigations reported here, one in 2016 (week 44) and 5 in 2018 (every 10th week from 10 to 50). For a formal representation of the problem, refer to [Claudet *et al.*, 2022], [Goh *et al.*, 2022], or [Guillaume *et al.*, 2022]. The 2018 weeks are publicly available as the *SatNet* benchmark dataset [Goh *et al.*, 2022].

2.2 Oversubscription

The DSN is routinely oversubscribed by a variable amount [Johnston and Lad, 2018], depending on the mix of missions and their planned activities. For example, when a mission reaches its destination science target, such as a planet or asteroid, it can go from minimal DSN usage to nearly continuous coverage for some period, sometimes months or years. Other missions act as orbiting relay stations for ground-based rovers. When oversubscription reaches a level of as little as 10% it is equivalent to having one additional antenna's worth of demand to remove before a feasible schedule can be developed and published. Typical oversubscription levels range up to about 40%, with even higher spikes. Note that oversubscription is not just a function of *total* time in the week, but

also of the geometry of when missions are visible: for spacecraft that are in the same part of the sky, their total demand can still overwhelm available resources during certain times of the day when they are all visible.

2.3 Scheduling Objectives

Natural objectives for the DSN scheduling problem might seem to be to maximize the total number of requests or total time scheduled, or to maximize antenna usage. However, there are severe drawbacks to using these as objectives for this problem: several dozen individual missions or science users submit their requests, which are widely varying in both total time and individual request duration, and thus with a range of difficulty in fitting into the schedule. A solution that fills the schedule by satisfying requests from one mission at the expense of another would be rejected by DSN users. In extreme cases, missions could be entirely dropped from the schedule, with others having incrementally more of their requests satisfied – an unacceptable scenario.

This leads to the definition of objectives that are based on how well scheduling requests are met on a per-mission basis. Two cost (minimization) objectives are therefore defined, based on a measure of how *unsatisfied* each mission could be with the extent to which their requests are met in the final schedule (with an assumption of uniform preferences among each mission's requests, e.g. [Johnston, 2021]):

The overall RMS unsatisfied time fraction is defined as:

$$U_{rms} = \sqrt{\frac{1}{N} \sum_{i=1\ldots N} \left(\frac{T_{Ri} - T_{Si}}{T_{Ri}} \right)^2} \quad (1)$$

where i ranges over missions $1 \ldots N$, and T_{Ri} and T_{Si} are the tracking time requested and scheduled, respectively, for mission i.

While this will reflect an overall measure of unsatisfied requests, it does not distinguish the situation where one mission is "pushed out" of the schedule altogether ($T_{Si} = 0$), an unacceptable situation. For this purpose, a better metric is:

$$U_{max} = \max_{i \in \{1\ldots N\}} \left(\frac{T_{Ri} - T_{Si}}{T_{Ri}} \right) \quad (2)$$

which indicates the *worst case* (max) unsatisfaction of any individual mission. Here, a value of 1.0 means *no* requested time for that mission was included in the schedule. We can compare two schedules based on their values of U_{rms} and U_{max}. Intuitively, U_{rms} can be thought of as "spread the pain", and U_{max} as "no user left out".

3 CSP-HS: CSP-based Heuristic Scheduling for the DSN

The heuristic search algorithm CSP-HS described in this work is based on a stochastic multi-start hill-climbing approach, implemented in three computational phases (see Figure 2):

1. **Initial assignment**: perform a greedy initial assignment of all requests to some valid time and resource for that request, avoiding but allowing conflicts

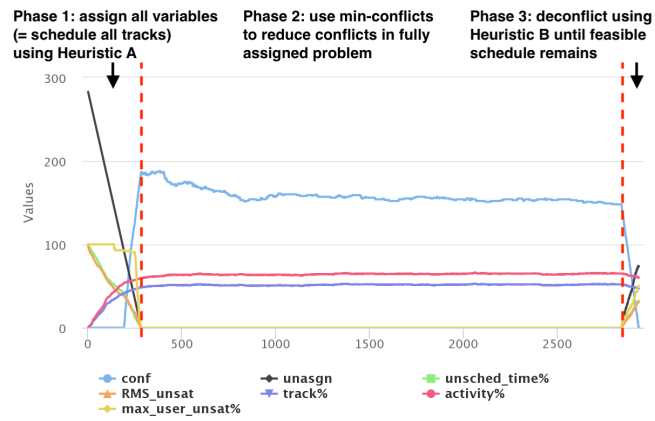

Figure 2: Illustration of the 3 phases of the CSP-HS algorithm, showing CSP runtime values as a function of step count. In the first phase, all variables are assigned, and when no conflict-free values are available, the conflict count grows. In the second phase, the min-conflicts heuristic is used to reduce conflicts by re-assigning variables with conflicted values. In the third and final phase, variables are unassigned until there are no conflicts remaining.

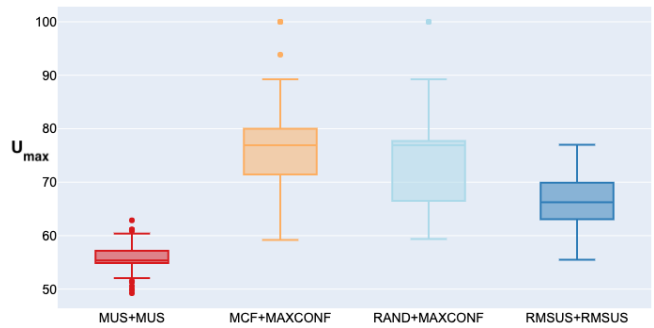

Figure 3: A comparison of four candidate heuristic pairs for phases 1 and 3, as assessed by the objective U_{max}. For this metric, the *max unsat* (MUS+MUS) combination performs best by far.

2. **Min-conflict hill climbing**: use a min-conflicts heuristic to repair the schedule and reduce conflicts, leaving all requests assigned [Minton *et al.*, 1992]
3. **Deconflict and repair**: remove conflicting activities and (selectively) add back others that may now have feasible places

In this approach, each phase can make use of different heuristics, described below, and we can assess the combinations that provide the most promising results and then evaluate them on larger data sets.

For an example run, Figure 2 shows the evolution of some quantities of interest during the three computational phases. These include:

- `conf`: the number of requests in conflict, i.e. violating any hard problem constraint
- `RMS_unsat`: the changing value of RMS user unsatisfaction U_{rms} (Eqn. 1)
- `max_user_unsat`: the changing value of the max user unsatisfaction U_{max} (Eqn. 2)

2016-2018 Problems

	2016 wk44			2018 wk10		2018 wk20		2018 wk30		2018 wk40			2018 wk50	
Metric	**ΔMILP**	**CSP-HS**	**QUBO +Leap**	**ΔMILP**	**CSP-HS**	**ΔMILP**	**CSP-HS**	**ΔMILP**	**CSP-HS**	**ΔMILP**	**CSP-HS**	**QUBO +Leap**	**ΔMILP**	**CSP-HS**
Hours satisfied	901 of 1418	950	**976**	822 of 1192	**860**	**1059 of 1406**	1034	983 of 1464	**998**	949 of 1737	984	**1059**	816 of 1292	**838**
Overall satisfied time fraction (%)	63.5	67.0	**68.8**	69.0	**72.2**	**75.3**	73.6	67.1	**68.1**	54.6	56.6	**61.0**	63.1	**64.9**
# satisfied requests	208 of 284	231	**245**	203 of 257	**224**	249 of 294	**261**	231 of 293	**248**	223 of 333	250	**269**	197 of 275	**224**
average satisfied request fraction (%)	73.2	81.3	**86.3**	79.0	**87.2**	84.7	**88.8**	78.8	**84.6**	67.0	75.1	**80.9**	71.6	**81.5**
Average satisfied fraction (%)	69.9	76.8	**82.8**	81.5	**84.5**	**88.8**	84.9	**81.4**	79.0	70.8	72.7	**80.3**	73.8	**77.8**
RMS Unsat (%)	35.4	**30.2**	31.5	26.5	**23.1**	**21.5**	21.8	29.4	**27.7**	40.5	35.6	**33.5**	34.9	**30.1**
Max Unsat User (%)	54.2	**49.2**	89.2	47.9	**44.2**	64.1	**50.3**	64.3	**52.9**	100	**60.1**	94.6	60.0	**54.4**
Runtime (hours)	~24	~1*	~2.5											

Figure 4: Summary of tabulated results comparing different approaches. Better values of the schedule metrics are indicated in bold font, and for fairness metrics U_{max} and U_{rms}, shaded in green as well. Runtime for CSP-HS is based on 100 (serial) runs, as described in the text.

- `unasgn`: number of unassigned variables, i.e. the number of requests without track assignments
- `track%`: percentage of tracks scheduled
- `unsched_time%`: percentage of time requested that is not scheduled
- `activity%`: percentage of activities scheduled

The underlying representation of the scheduling problem is based on discretizing the feasible times throughout the week of interest, initially using a time quantum of 15 minutes (see discussion below). Variables correspond to requests, while the decision value x_j for each request j encapsulates simultaneously a choice for start time, duration, and antenna: (t_j, d_j, A_j). For requests with splittable tracks, this is extended to include when to split, and then each split segment start time, duration, and antenna. For multiple antenna requests, it also includes the set of antennas (the start time and duration must be the same). Any choices that can be excluded (not in view, antenna not available, etc.) are omitted from the model.

During all three phases it is necessary to keep a tally of how many variables and their values are placed in conflict due to other variable assignments. For example, scheduling a simple request at a specific time would place any other overlapping requests on the same antenna in conflict, and so would increment the conflict count on all those value choices. Conflicts are calculated on the fly and then cached, so it is fast to undo/redo any trial assignment.

Both the initial assignment (1) and the deconfliction phase (3) are driven by heuristics that are applied to the then current state of the model. Several heuristics were explored to determine which performed best in the context of objectives related to fairness, focusing on U_{max} (see Figure 3). Some of the variable (request) selection heuristics assessed are listed below; value selection is random from the set of min-conflict values:

- MUS: most unsatisfied user – the request that can contribute most to improving U_{max}
- RMSUS: the request that can contribute most to improving U_{rms}
- MCF: most constrained first – the request with the fewest min-conflict value choices
- MAXCONF: the request with an assignment that has largest number of conflicts
- RAND: a random request

4 Results

CSP-HS was run on each of the 6 test problems (best of 100 runs), with the results tabulated in Figure 4 and plotted in Figure 5. All of the problems were also solved by Δ-MILP [Claudet *et al.*, 2022], and two of them were solved by QUBO+Leap [Guillaume *et al.*, 2022]. No results were reported by the RL investigation.

The solution metrics reported in Figure 4 are:

- Hours satisfied: total hours scheduled as compared with total requested for the week
- Overall satisfied time fraction (%): number of hours satisfied as a fraction of the total requested
- # satisfied requests: total number of requests successfully scheduled as compared with total for the week
- Average satisfied request fraction (%): number of requests satisfied as a fraction of the total
- Average satisfied fraction (%): the average over all users i of the satisfied request fraction, T_{Si}/T_{Ri}
- RMS user unsatisfaction U_{rms} (%) - Equation 1
- Max user unsatisfaction U_{max} (%) - Equation 2
- Runtime (hours): approximate total runtime for 2016wk44 (other problems are similar and not reported)

4.1 Solution Quality

In Table 4, better values are bolded and the best values of the fairness metrics U_{rms} and U_{max} are also highlighted in green. One of the results evident from the table is that it is common for methods to "pack" the schedule with smaller and easier-to-schedule activities, with the tradeoff being that some missions have their time reduced, in order to fit in more activities for others. This highlights the difficulty of accommodating the fairness metrics that ensure no mission is starved. In contrast, CSP-HS does a much better job on U_{max} while still populating the schedule to a very high level. In only two problems did other methods do slightly better in U_{rms} and, in both cases, the max user unsatisfaction metric U_{max} was much worse (by 14 percentage points or more).

Of the 6 test problems, 2018wk40 is the most oversubscribed: it has the largest number of requests and of requested tracking hours, and the smallest fraction of requested time scheduled. This is a consequence of the activities planned around that time: one mission approaching Mars for landing, and two others approaching their asteroid targets, with all of these events requiring much higher tracking and communications coverage than at other times. For this week, both Δ-MILP and QUBO-Leap had U_{max} values >95%, while CSP-HS came in at a much lower level of 60%. At this level of oversubscription, it is essential to balance the mission set so that in spite of very heavy demand, no users are dropped out of the schedule. In this particular problem, what is observed in the solutions generated by Δ-MILP and QUBO-Leap is that several large users are massively under requirement (5% or less), in order to build up the scheduled time of the others. Such schedules are completely unacceptable in real DSN operations.

4.2 Runtime Performance and Scalability

As a multi-start stochastic algorithm with a hill-climbing phase, CSP-HS is run for a specified number of iterations. Empirically, phase (2) is run for $10 \times K$ iterations where K is the total number of requests. Experiments with other values have been run, and it has generally been found that improvements plateau after this point. As with the number of iterations, the total number of runs is also specified and has been set to 100 for this experiment. Note that these runs are completely independent and so could be trivially parallelized, but the timing reported here is the serial runtime as a worst case.

Typical CSP-HS runtimes (100 runs, $10 \times K$ min-conflicts iterations, on an Intel Core i9 MacBook Pro) are close to 1 hour (serialized, 35 seconds/iteration). QUBO-Leap runs in roughly 2.5 hours, which is split between QUBO problem generation and running the solver. Both CSP-HS and QUBO-Leap could benefit from parallelization. Δ-MILP runs over a time range from 7.5 to 22.5 hours, with the most oversubscribed schedules taking longest. While the actual step in the DSN scheduling process in which this algorithm would run allows for such long runtimes, it is clearly advantageous to run quickly to allow for iterations. As usual, there is a caveat on comparing runtimes across the different methods: each method ran in development/test environments appropriate to their methodologies, and these environments do not reflect optimizations that would eventually be expected.

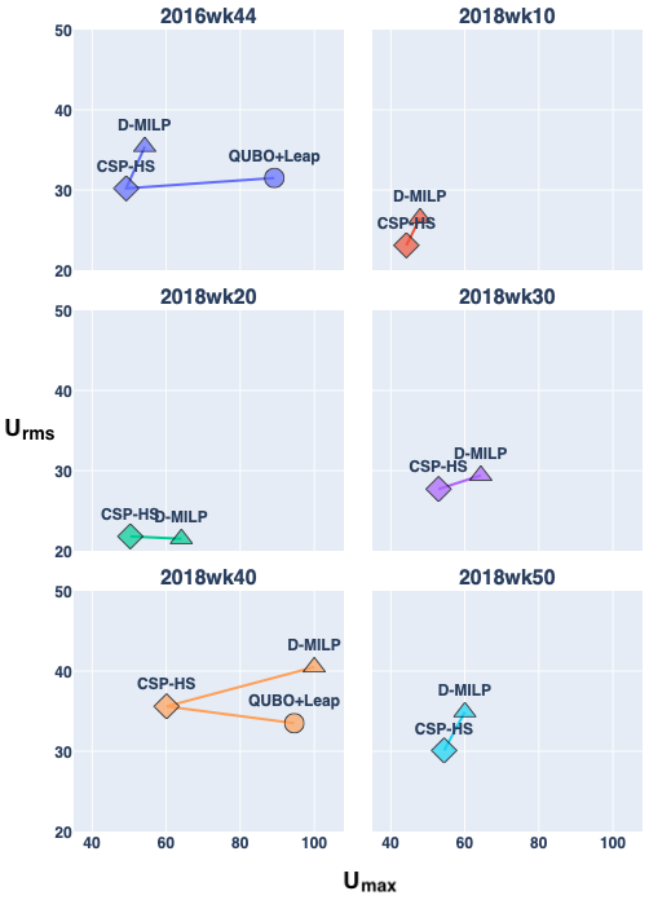

Figure 5: Summary of results for the 6 test weeks, plotted vs two key minimization objectives, U_{max} and U_{rms} – better results are to the lower left. Results are labeled by the method used (see table). CSP-HS is in all cases better in U_{max}; in two problems, 2018w20 and w40, other methods perform slightly better in U_{rms}, but only by starving one or more users to improve the overall RMS (at the expense of worse values of U_{max}.)

All method results reported in Figure 4 were run with 15-minute time granularity. In actuality, the DSN runs on schedules with 5-minute granularity, and so additional tests of CSP-HS were run with this setting. End-to-end wall clock time per run increased from 35 seconds (15-minute granularity) to 291 seconds (5-minute granularity), a factor of 8.3×, corresponding to $O(T^2)$ in the number of time points T. With full parallelization, CSP-HS solutions could be generated in about 5 minutes. No other methods reported solutions with 5-minute granularity.

5 Conclusions and Next Steps

We have described a heuristic search framework CSP-HS for solving heavily oversubscribed scheduling problems such as those found in NASA's Deep Space Network, where an important part of any solution is the satisfaction of "fairness" metrics for the allocation of time. The CSP-HS framework lends itself to "pluggable" heuristics, depending on the key objectives that apply. For the DSN, solutions found for a

sample problem set are both higher quality and found more quickly than all reporting comparison methods, including a Quantum Annealing (QA) QUBO-based method using a hybrid quantum solver [Guillaume *et al.*, 2022] and the Δ-MILP approach based on a Mixed-Integer Linear Programming formulation [Claudet *et al.*, 2022].

Fairness often plays a major role in problems like this (e.g. [Kash *et al.*, 2014]), in that it is essential not to starve one user to accommodate others. In this particular problem, fairness is reflected by the allocation of time in proportion to that requested, due to an upstream process that validates requested levels as acceptable. Other aspects of the problem could contribute – for example, there could be a threshold by mission for minimum acceptable allocations. The approach presented here can contribute to solution options in these kinds of cases, in that heuristics can be incorporated directly into the CSP-HS runtime phases.

While the sample problems discussed above capture the driving characteristics of the initial phases of DSN scheduling, there are some significant elements of the full-scale DSN scheduling problem that are not included. Chief among these are the following:

- time separation between tracks: for many missions with regular tracking intervals, tracks must be separated by some minimum and/or maximum time from adjacent tracks
- multiple spacecraft per antenna (MSPA): for missions that are close enough in the sky to be in the antenna beam at the same time (e.g. the missions at Mars), DSN supports multiple downlinks at one time (though only a single uplink). This is an efficiency that is heavily used in DSN operations and needs to be incorporated in the schedule.

Another area of ongoing work is related to priorities and preferences: requests submitted by users are not all equal to them, and work has been proceeding on including priorities (such as for critical events like maneuvers, orbit insertions, planetary landings) as well as user-specified preferences. Both will affect scheduling in that a better schedule will be one that satisfies higher priority and higher preference requests, while respecting the fairness metrics described above [Johnston, 2021]. Implementation of this in DSN operations is underway.

Acknowledgements: This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[Claudet *et al.*, 2022] Thomas Claudet, Ryan Alimo, Edwin Goh, Mark D. Johnston, Ramtin Madani, and Brian Wilson. Δ-MILP: Deep Space Network Scheduling via Mixed-Integer Linear Programming. *IEEE Access*, 10:41330–41340, 2022.

[Goh *et al.*, 2021] Edwin Goh, Hamsa Shwetha Venkataram, Mark Hoffman, Mark D. Johnston, and Brian D. Wilson. Scheduling the NASA Deep Space Network with Deep Reinforcement Learning. In *Proceedings IEEE Aerospace Conference*, Big Sky, MT USA, 2021.

[Goh *et al.*, 2022] Edwin Goh, Hamsa Shwetha Venkataram, Bharathan Balaji, Brian D. Wilson, and Mark D. Johnston. SatNet: A Benchmark for Satellite Scheduling Optimization. In *AAAI-22 Workshop on Machine Learning for Operations Research (ML4OR)*, 2022.

[Guillaume *et al.*, 2022] A. Guillaume, Edwin Goh, Mark D. Johnston, Brian Wilson, Anita Ramanan, Frances Tibble, and Brad Lackey. Deep Space Network scheduling using Quantum Annealing. *IEEE Transactions on Quantum Engineering*, page submitted, 2022.

[Imbriale, 2003] William A. Imbriale. *Large Antennas of the Deep Space Network*. Wiley, 2003.

[Johnston and Lad, 2018] Mark D. Johnston and Jigna Lad. Integrated Planning and Scheduling for NASA's Deep Space Network – from Forecasting to Real-time. In *SpaceOps 2018*, Marseilles, France, 2018.

[Johnston *et al.*, 2014] Mark Johnston, Daniel Tran, Belinda Arroyo, Sugi Sorensen, Peter Tay, John Carruth, Adam Coffman, and Mike Wallace. Automated Scheduling for NASA's Deep Space Network. *AI Magazine*, 35:7–25, February 2014.

[Johnston, 2019] Mark D. Johnston. User Preference Optimization for Oversubscribed Scheduling of NASA's Deep Space Network. In *11th International Workshop on Planning and Scheduling for Space (IWPSS)*, pages 86–92, July 2019.

[Johnston, 2020] Mark D. Johnston. Scheduling NASA's Deep Space Network: Priorities, Preferences, and Optimization. In *Proceedings ICAPS SPARK Workshop*, Nancy, France (virtual), October 2020.

[Johnston, 2021] Mark Johnston. User Preference Optimization for Oversubscribed Scheduling of NASA's Deep Space Network. In *SpaceOps*, Capetown (Virtual), 2021.

[Kash *et al.*, 2014] Ian Kash, Ariel Procaccia, and Nisarg Shah. No Agent Left Behind: Dynamic Fair Division of Multiple Resources. *Journal of AI Research*, 51:579–603, 2014.

[Minton *et al.*, 1992] Steven Minton, Mark D. Johnston, A. Philips, and P. Laird. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence Journal*, 58:161–205, 1992.

[Sabol *et al.*, 2021] Alex Sabol, Ryan Alimo, Farhad Kamangar, and Ramtin Madani. Deep Space Network Scheduling via Mixed-Integer Linear Programming. *IEEE Access*, 9:39985–39994, 2021.

[Shouraboura *et al.*, 2016] Caroline Shouraboura, Mark D. Johnston, and Daniel Tran. Prioritization and Oversubscribed Scheduling for NASA's Deep Space Network. In *ICAPS SPARK Workshop*, London, 2016.