SatNet: A Benchmark for Satellite Scheduling Optimization

Edwin Goh[1], Hamsa Shwetha Venkataram [1], Bharathan Balaji [2], Mark D Johnston [1], Brian Wilson [1]

[1]Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California 91109
[2] Amazon*
Seattle, Washington 98109
[1]edwin.y.goh@jpl.nasa.gov, [1]venkatar@jpl.nasa.gov, [2]bhabalaj@amazon.com, [1]mark.d.johnston@jpl.nasa.gov,
[1]bdwilson@jpl.nasa.gov

Abstract

Satellites provide essential services such as networking and weather tracking, and the number of near-earth and deep space satellites are expected to grow rapidly in the coming years. Communications with terrestrial ground stations is one of the critical functionalities of any space mission. Satellite scheduling is a problem that has been scientifically investigated since the 1970s. A central aspect of this problem is the need to consider resource contention and satellite visibility constraints as they require line of sight. Due to the combinatorial nature of the problem, prior solutions such as linear programs and evolutionary algorithms require extensive compute capabilities to output a feasible schedule for each scenario. Machine learning based scheduling can provide an alternative solution by training a model with historical data and generating a schedule quickly with model inference. We present SatNet[1], a benchmark for satellite scheduling optimization based on historical data from the NASA Deep Space Network. We propose formulation of the satellite scheduling problem as a Markov Decision Process and use reinforcement learning (RL) policies to generate schedules. The nature of constraints imposed by SatNet differ from other combinatorial optimization problems such as vehicle routing studied in prior literature. Our initial results indicate that RL is an alternative optimization approach that can generate candidate solutions of comparable quality to existing state-of-the-practice results. However, we also find that RL policies overfit to the training dataset and do not generalize well to new data, thereby necessitating continued research on reusable and generalizable agents.

Introduction

Driven by advances in space exploration, particularly with the imminent plans for lunar exploration, the space communications sector is growing rapidly. Communication of satellites with ground stations is essential to downlink valuable data and uplink mission-critical commands. Scheduling of satellite communications can be challenging as satellites can request more bandwidth than is available and are visible for limited periods from the ground station. The number of possible schedules grows combinatorially with the number of satellite requests and the antennas available. We study the scheduling optimization problem so that the ground station resources are fully utilized while ensuring fair allocation to satellite requests.

We use the NASA Deep Space Network[2] (DSN) as an example system for satellite scheduling optimization problems. A number of algorithms have been proposed for scheduling of the DSN (Clement and Johnston 2005; Johnston 2008; Johnston et al. 2014; Johnston 2020; Sabol et al. 2021), but they primarily rely on compute intensive heuristics due to the NP-hard nature of the problem. We explore data driven methods to solve the satellite scheduling problem given success in combinatorial optimization problems such as vehicle routing (Nazari et al. 2018) and bin packing (Balaji et al. 2019; Mazyavkina et al. 2021). Existing solutions require extensive computation for each instance of the problem. On the other hand, a machine learning (ML) model can learn the distribution of communication requests from historical data, and the resulting policy can generate optimized schedules with a neural network inference. A stochastic policy can generate a diverse set of candidate schedules, and it is possible to incorporate additional objectives such as fairness in the reward function.

To our knowledge, there are no datasets available to train ML models or benchmark existing algorithms in this domain. We present SatNet, a dataset consisting of historical communications requests by interplanetary spacecraft using the NASA DSN and their corresponding visibility to antennas on the ground stations. We formulate the scheduling optimization problem as a Markov Decision Process (MDP) with the objective of satisfying the maximum number of hours of communication requested by spacecraft. We train policies using the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017), and the resulting schedules are competitive with those from state-of-the-practice heuristics. However, we find that generalization of the policies to new data is challenging, and identify avenues of further research in both problem formulation and algorithm development.

Related Work

Satellite scheduling is an important problem that has been studied since the earliest space missions, and with the onset

*work unrelated to Amazon

[1]SatNet is available at https://github.com/edwinytgoh/satnet

[2]https://eyes.nasa.gov/dsn/dsn.html

of low-cost satellites like smallsats, CubeSats, the scheduling task is becoming increasingly challenging because the level of available resources (e.g., antennas, expert human schedulers) is unable to keep up. In their expansive survey, (Xhafa and Ip 2021) highlight scheduling problem as central to satellite mission planning for effective communication between operations teams. Works such as (Prins 1994), (Tangpattanakul, Jozefowiez, and Lopez 2015), (Wong et al. 2016) and (Lee et al. 2002) propose various algorithms and methods to the problem of scheduling under different use cases involving CubeSats, satellite multimedia networks etc. On the other hand, there have been targeted efforts that specifically propose solutions to the DSN scheduling problem using deep reinforcement learning (Goh et al. 2021), evolutionary computational methods (Guillaume et al. 2007), heuristic search techniques (Johnston 2020), and mixed integer linear programming (Sabol et al. 2021; Claudet et al. Forthcoming). Nonetheless, as outlined earlier, related research in this field use custom built datasets that are specific to each task. Since the key to creating an open ML research community is a benchmark, this is the gap that our present work hopes to bridge through the SatNet dataset.

Interplanetary Communications Scheduling

Efficient scheduling is especially relevant for interplanetary science missions because an increasing number of interplanetary spacecraft need to be served by a fixed set of operational facilities powerful enough for interplanetary communications. NASA's DSN is a major provider of such capabilities, and is frequently oversubscribed. DSN communicates with $\sim$30 interplanetary spacecraft active at any given time.

Problem Elements and Terminology

Satellite communications scheduling problems pose a few unique challenges and constraints that are different from those associated with terrestrial communications scheduling. In this section, we define the relevant concepts and terminology used throughout SatNet.

The main elements of a request are the **spacecraft** involved, the different **missions** m associated with the spacecraft, and the **antenna(s)** or ground station on which communications is established.

In DSN, spacecraft communicate with ground stations located in one (or more, in the case of multi-antenna requests) of the three DSN complexes in Goldstone, Canberra, and Madrid. These complexes operate antennas, which are also known as **resources** [3].

Since antennas can only communicate with spacecraft that are visible in the sky, transmissions can only occur during *view periods* — periods during which satellites are visible to the antennas. View periods v are defined for each spacecraft-resource pair based on the spacecraft's orbital elements. They are bounded by v^b and v^e, where b and e mean "beginning" and "end" of the view period, respectively.

In addition to physical constraints on visibility, transmission windows are also constrained by operational considerations that are specific to each mission. These time window constraints are associated with each request r, and are used to determine the set of *usable* or *valid* view periods for each request. View periods that fall outside the specified time windows are considered unusable. This constraint on visibility windows is similar to time window constraints encountered in other operations research problems such as the Vehicle Routing Problem with Time Windows (VRPTWs) (Kolen, Rinnooy Kan, and Trienekens 1987).

Problem Elements Summary

In summary, the scheduling problem consists of a set of N requests, $\mathcal{R} = \{r_i \mid i = 1, 2, ..., N\}$. In turn, each request r_i contains:

1. A mission corresponding to a spacecraft, $m_i \in \mathcal{M}$
2. A set of A antennas/resources that can be used to fulfill this request, $\mathcal{A}_i = \{a_{i,j} \mid j = 1, 2, ..., A\}$
3. A set of V view periods during which m_i is visible to $a_{i,j}$, $\mathcal{V}_{i,j} = \{[v^b_{i,j,k}, v^e_{i,j,k}] \mid k = 1, 2, ..., V\}$
   Note that there are A sets of view periods, each of potentially different sizes, including the null set.
4. A minimum requested duration, $d_{min,i}$
5. A nominal requested duration, d_i
6. A time window constraint that specifies when a mission wants transmissions to take place, $[t^s_i, t^e_i]$.

Here, we assume that each mission operates one spacecraft. Multi-spacecraft (i.e., constellation) missions can be treated using an additional index, or by treating each spacecraft as a separate mission. DSN request sets currently adopt the latter approach.

The total requested duration in a given week is given as,

$$T_R = \sum_{i}^{N} d_i \tag{1}$$

Scheduling Objective

Satellite scheduling is inherently a multi-objective optimization problem. Ground station operators need to maximize satisfaction of communications requests under capacity constraints. The extent to which each user's requests is fulfilled presents another optimization objective, which is the notion of fairness across users. These high-level objectives can be expressed in various different forms, which include:

1. Maximizing the total hours scheduled across all antennas
2. Maximizing the number of satisfied requests across users
3. Minimizing the time window conflicts between users
4. Maximizing the utilization/duty cycle across all antennas
5. Maximizing the total hours scheduled on missions with the fewest hours allocated (i.e., maximin)

[3] Note that in DSN terminology, resources represent a more general abstraction of the antenna; they are in fact the *combination* of antennas with which spacecraft can communicate. Except for multi-antenna/arrayed requests, this combination of antennas is generally of size 1. Thus the term 'resources' is frequently used interchangeably with 'antennas'.

6. Minimizing the variance between fraction of requested time satisfied for each user

Indeed, there are numerous other creative representations in service of the high-level scheduling objectives of maximum request satisfaction in a fair manner. The relative importance of the objectives also depend on the specific space application. For instance, a commercial ground station operator might choose to prioritize high-value requests (thus sacrificing fairness), while a NASA ground station might prioritize fairness to maximize overall mission success across all federal investments.

In the proposed RL approach, we consider the single objective problem of maximizing the total number of scheduled hours across all users. While not operationally practical, the single-objective problem provides a reasonable starting point for diagnostics and analysis of the approach.

Fairness Objectives The baseline MILP and RL results include a metric U_{RMS}, which is the root mean square of the unsatisfied time fraction across the set of all missions, $\mathcal{M}$. The unsatisfied time fraction of a mission m is given as,

$$U_m = \frac{T_{R,m} - T_{S,m}}{T_{R,m}} \tag{2}$$

where $T_{R,m}$ and $T_{S,m}$, the total requested and satisfied durations for mission m, respectively, are given as

$$T_{R,m} = \sum_{i=1}^{N} d_i \Big|_{m_i=m} \tag{3}$$

$$T_{S,m} = \sum_{i=1}^{N} |T_{i,j}| \Bigg|_{m_i=m} \tag{4}$$

where $T_{i,j}$ is a track for request r_i that is allocated on antenna $a_{i,j}$. Note that the unsatisfied time fraction is 1 minus the satisfied time fraction. The root mean square of this quantity, U_{RMS}, represents the standard deviation of the unsatisfied time fraction across all missions. U_{RMS} is one way to capture the "fairness" of a schedule (Johnston 2020); a schedule with a lower effective unsatisfied time fraction on average would have a lower U_{RMS}, and vice versa. U_{RMS} is given as

$$U_{RMS} = \sqrt{\frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} (U_m)^2} \tag{5}$$

While U_{RMS} captures the overall unsatisfied time fraction and the level of variation amongst different missions, it remains less sensitive to extreme cases of unfairness. For example, if no communications time is allocated for any request from a given mission, i.e., $T_{S,m} = 0$, the U_{RMS} metric would obviously increase. However, this increase does not capture the severity of neglecting an entire mission. In these and other situations, a complementary metric, U_{MAX}, can be used to identify the mission with the highest unsatisfied time fraction. U_{MAX} is given by Eq. 6 and captures the same idea as item 5 in the preceding discussion on different objectives.

$$U_{MAX} = \max_{m \in \mathcal{M}} U_m \tag{6}$$

Typical Scheduling Constraints

In addition to the visibility and time window constraints, the scheduling problem is also subject to resource and operational constraints. We summarize the constraints as follows:

1. Each resource can only communicate with one spacecraft at a given time. In other words, there may be no overlaps between activities on any ground station[4].
2. A scheduled track for the i^{th} request on resource $a_{i,j}$, $T_{i,j}$, must fall within a valid view period. Formally, $T_{i,j} \subseteq \mathcal{V}_{i,j}$. Recall that a valid view period is one that falls within the user-specified time window for a particular request.
3. A scheduled track's duration must meet the minimum requested duration $d_{min,i}$ and must not exceed the requested duration d_i.
4. For requests that require simultaneous access to multiple ground stations[5], tracks must fall within view periods that overlap across all ground stations.

Furthermore, near-Earth communications applications have constraints related to the hand-off between one ground station and another. These include duration limits on communications gaps as well as the timing of those gaps so as to maintain Quality of Service requirements or ensure mission safety.

Setup and Teardown Times Tracks on the DSN must have the correct *setup* and *teardown* durations. These operational segments are appended to the start and end of transmission, respectively, and allow for ground station calibration and pointing. Because no transmissions occur during the setup and teardown phases, these segments of the track are allowed to fall outside $\mathcal{V}(i,j)$, provided that antenna j is available for the entire setup and/or teardown duration. The required setup and teardown durations are provided for each request.

Splitting One of the unique aspects of DSN scheduling is the relatively long transmission duration compared to near-Earth satellite communications. This is primarily due to the overall greater distance of DSN users' spacecraft from Earth, which reduces throughput because of attenuation but increases the duration a spacecraft is visible to a ground station. This enables long-running communications requests to be split into two or more segments. In the DSN scheduling problem, the primary constraints on splitting are as follows:

1. Splitting is allowed for requests that are longer than or equal to 8 hours.
2. Each split track needs to be at least 4 hours long

[4]Recent technological upgrades have enabled the NASA Deep Space Network to offer Multiple Spacecraft per Antenna (MSPA) operations for Mars missions, which relaxes this constraint for a subset of missions. For generality, this is not reflected in present results.

[5]e.g., for Time Division Multiple Access (TDMA) in telecommunications applications, or in the case of the DSN, Delta-Differential One-way Ranging (DDOR) and signal aggregation across multiple antennas to increase signal strength

3. Each split track must fall within a valid view period
4. The total duration of all split tracks must meet the minimum requested duration $d_{min,i}$ for the original request, and not exceed the requested duration d_i.
5. Each split track must have its own setup and teardown segments, which are allowed to fall outside of valid view periods as long as the antenna is available.

Antenna Maintenance Most ground stations undergo regular maintenance to enable reliable operation. Some may have flexible maintenance schedules in which maintenance is performed whenever there are gaps between tracking activities, while other applications may have rigid schedules that have higher priority than regular communications. We adopt the latter approach because historical maintenance data are readily available, and because the reduced antenna vacancy presents a more challenging optimization problem.

The SatNet Benchmark Dataset

The SatNet dataset consists of mission requests and corresponding DSN antenna availability for 5 oversubscribed weeks in 2018. The relevant metrics include the number of total requests, N, total requested hours, T_R, number of missions, and number of usable antennas. Note that because requests can use combinations of antennas, the number of usable *resources* will be higher than the number of usable antennas (12 in this time period). These metrics are summarized in Table 1. The dataset is a single JSON file with entries for each of the five weeks. A week entry is further broken down into a list of requests, where each request contains the key elements contained in the Problem Elements Summary section.

In addition to the problem JSON file, SatNet also includes a maintenance history across all 12 antennas in 2018. The maintenance history is represented as a CSV file with the corresponding antenna, start times and end times.

Table 1: DSN subscription level varies by the time of year. The table below depicts request distributions from the five weeks from year 2018 that are provided in SatNet.

Week	# Usable antennas	# Requests, N	Total requested hours, T_R	# Missions
10	12	257	1192	30
20	12	294	1406	33
30	12	293	1464	32
40	12	333	1737	33
50	12	275	1292	29

Problem Formulation

We implement a satellite scheduling simulator using the OpenAI Gym API.

State Space

Our RL formulation leverages a state space represented as a 1D vector. This state vector provides the state of the schedule, including the number of missions, requests and requested hours remaining (1x3), the remaining duration of each request (1xN), and the remaining number of hours available on each antenna (1x15). Thus, the dimensionality of the state space is (1xN+18), where N is the number of requests (typically ~ 300),

$$\mathbb{S} \in \mathbb{R}^{N+18} \tag{7}$$

The simple 1D state space facilitates a “batch” approach to satellite scheduling by providing the entire set of requests to the agent at once, as opposed to presenting requests to the agent in sequential fashion. This batch scheduling approach is inspired by the canonical knapsack problem in combinatorial optimization. The knapsack problem involves choosing from a set of n items (of prescribed weight and value) to include in a knapsack of fixed size such that the overall value is maximized.

Action Space

The action space $\mathbb{A} = \{1, ..., N\}$, not to be confused with the set of antennas $\mathcal{A}_i$, is the set of integers $\{1, ..., N\}$. An action at time t, $a_t \in \mathbb{A}$, represents the index of a particular request in the environment's internal array representation. For example, $a = 2$ corresponds to the third request in the array.

State Transition

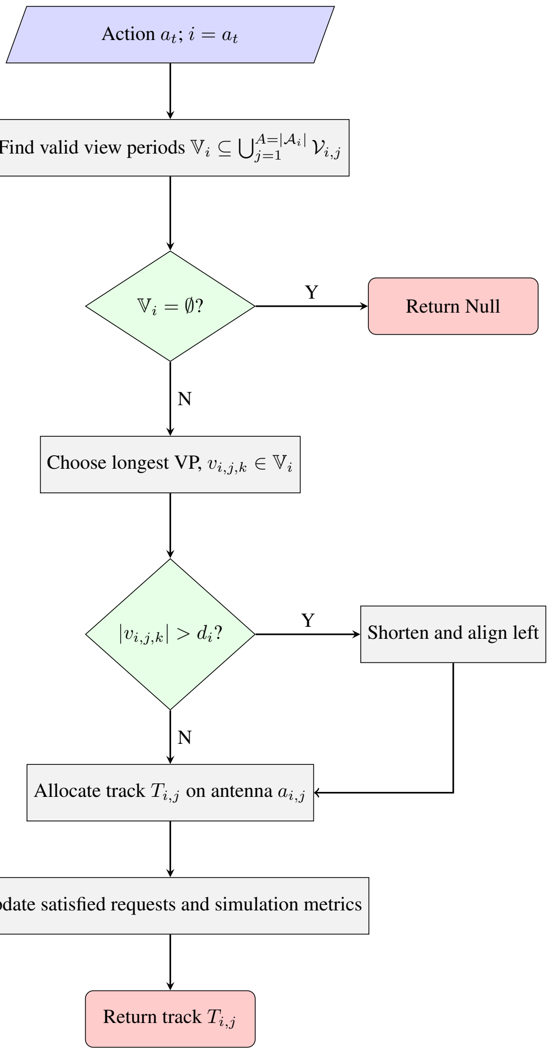
The environment's state transition/simulation step is illustrated in Fig. 1. Given an action, $a_t \in \mathbb{A}$, we assign the request index, $i = a_t$. The environment retrieves the usable antennas $\mathcal{A}_i$ and the set of valid view periods, $\mathbb{V}_i \subseteq \bigcup_{j=1}^{A=|\mathcal{A}_i|} \mathcal{V}_{i,j}$. If there are valid view periods, i.e., $\mathbb{V}_i \neq \emptyset$, the simulator chooses a view period, $v_{i,j,k} \in \mathbb{V}_i$, based on a prescribed greedy heuristic that simply selects the longest view period from $\mathbb{V}_i$. Next, the simulator shortens/clips $v_{i,j,k}$ if it is longer than the requested duration d_i, or if splitting is requested for a specific track. This shorten function takes a heuristic that tells the environment whether to align the resulting $T_{i,j}$ to the left, center, or right of $v_{i,j,k}$. Finally, the simulator allocates the potentially shortened $v_{i,j,k}$ onto the corresponding j^{th} antenna for this request, i.e., $a_{i,j}$.

Reward

The challenge in defining a reward function is analogous to the earlier discussion on scheduling objectives — there are many ways in which one can formulate the reward in an attempt to achieve the end goal of a fair and optimally satisfied schedule. For our initial results, we treat the single-objective problem of maximizing the total number of hours scheduled in a given week. The environment returns at the conclusion of each step the track duration that was allocated as a result of the agent's action, $r_t = d_t = |T_{i,j}|$.

Thus the total reward or *return*, R, for an episode is equal to the total scheduled duration, T_S. That is,

$$R = T_S = \sum_{t=0}^{N_t} d_t(s_t, a_t) \tag{8}$$



Action a_t; $i = a_t$
Find valid view periods $\mathbb{V}_i \subseteq \bigcup_{j=1}^{A=|\mathcal{A}_i|} \mathcal{V}_{i,j}$
$\mathbb{V}_i = \emptyset$?
Y
Return Null
N
Choose longest VP, $v_{i,j,k} \in \mathbb{V}_i$
$|v_{i,j,k}| > d_i$?
Y
Shorten and align left
N
Allocate track $T_{i,j}$ on antenna $a_{i,j}$
Update satisfied requests and simulation metrics
Return track $T_{i,j}$

Figure 1: Processes in the scheduling simulator's state transition. These steps are hidden from the agent, which only receives signals through states (remaining request distribution) and rewards (hours allocated) in between each action.

Episode Termination

An episode ends if:

1. all requests have been satisfied (an unlikely scenario); or
2. there are no more valid positions on the antennas to accommodate the remaining requests; or
3. the simulation exceeds a prescribed number of timesteps.

Request Shuffling

The simulation provides an option for users to toggle shuffling of the requests to prevent the agent from memorizing the order of requests after numerous episodes. When shuffling is enabled, the internal array representation is fixed to reduce computational cost. Instead, we shuffle a list of each request's unique identifier, or ID. With shuffling enabled, the previous example $a = 2$ points to the third ID in the shuffled list, which is then mapped back to a particular row in the array representation. This may or may not correspond to the second row in the array. To prevent "confusing" the agent, the state space is also re-ordered to match the order of the shuffled IDs such that successfully allocating the third ID corresponds to a reduction in remaining duration in the third element of the observation.

Enforcing Constraints

The RL formulation primarily uses action masking to ensure constraint satisfaction. As part of the observation returned at the end of each simulation step, the environment also returns a boolean array of size N that masks out requests that have been satisfied and requests that no longer have valid view periods as a result of other requests occupying antenna time. This mechanism prevents the agent from selecting invalid requests, thereby preventing wasted steps with no reward, which decreases the total runtime for each episode.

One of the advantages of the simple approach is to have the scheduling simulator handle all constraints in the state transition, encapsulating or hiding them from the agent. The majority of these scheduling constraints come into play when finding $\mathbb{V}_i$ and allocating track $T_{i,j}$ on antenna $a_{i,j}$. Future formulations can provide richer representations in the observation space, with information that agents can use to learn the scheduling constraints by penalizing invalid actions in the reward function.

MILP Baselines

We use a mixed integer linear programming (MILP) formulation of the DSN scheduling problem (Claudet et al. Forthcoming) as the baseline for SatNet. This formulation is itself comparable to expert-guided heuristic search algorithms, and takes into account the "fairness" metric by simultanouesly maximizing T_S, minimizing U_{RMS}, and minimizing U_{MAX}.

Table 2: Baseline MILP results from (Claudet et al. Forthcoming)

Week	T_S / T_R (hours)	U_{RMS}	U_{MAX}	Number of satisfied requests	Run time (hours)
10	822 / 1192	0.26	0.48	203	18
20	1059 / 1406	0.21	0.64	249	10.5
30	983 / 1464	0.29	0.64	231	13.5
40	949 / 1737	0.40	1.00	223	22.5
50	816 / 1292	0.35	0.60	197	7.5

RL Experiments

In this section, we briefly describe the experimental setup and ablation studies that were performed. We then present the initial results obtained using the RL formulation and compare the RL results to existing baselines. The experiments were run on a single Amazon EC2 cloud instance with 16 GPUs and 96 CPUs.

Training Regime

We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017) to train RL scheduling agents with a SGD minibatch size of 8192 and train batch size of 160000, along with tuned humanoid-v1 hyperparameters [6]. We used the large batch size to ensure sufficient exploration when interacting with the environment, and used a learning rate of 0.001. Convergence was defined to occur when the 50 most recent PPO iterations were within one hour from each other.

Results

RL results from the experiments described in the foregoing sections are presented for the SatNet dataset in Table 3. All results are presented with request shuffling turned off. After convergence, we run 1,000 episodes of inference to generate candidate schedules for each week. The numerous candidate schedules are possible because the learned policy is stochastic, and does not repeat the same action sequences each episode. From the candidates, we choose the solution with the highest T_S that has $U_{MAX} < 1$. Examples of the solution space spanned by the 1,000 candidate schedules are shown in Figs. 2 and 3, respectively.

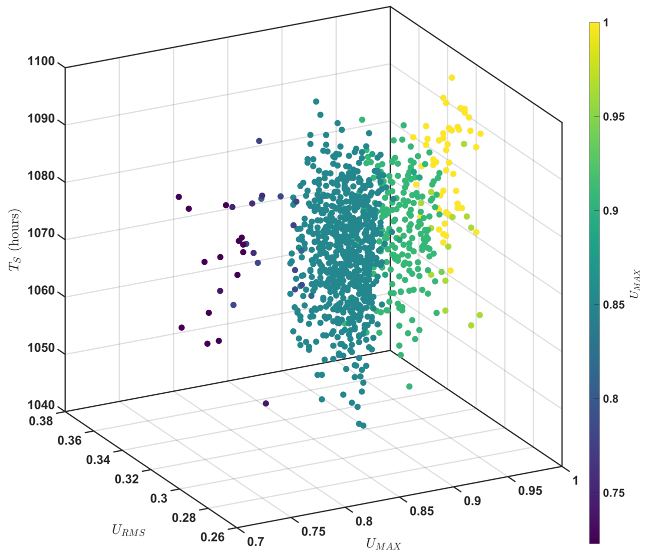


Figure 2: On week 30, the trained RL agent produced a best schedule of $T_S = 1{,}100$ hours, $U_{RMS} = 0.28$, $U_{MAX} = 0.85$ among 1,000 inference runs. Relevant metrics of the solution space covered by 1,000 RL inference runs on week 30 of the SatNet dataset. Points are colored according to U_{MAX}; darker is better.

Discussion

Effects of Shuffling

As described in earlier sections, the scheduling simulator provides the capability to shuffle the order of requests in the agent's observation space. The motivation behind shuffling the sequence of requests is to enable the agent to generalize to other problem weeks beyond the one on which it is trained. However, we found that the agent could only exhibit significant progress when we held constant the sequence of requests. This effect is shown in Fig. 4. This showed that the agent memorized the sequence of requests and overfit to the training dataset. On the contrary, shuffled requests did not help the agent generalize, and also did not maximise the mean reward even after extended periods of training (36 hours).

[6]https://github.com/ray-project/ray/blob/master/rllib/tuned_examples/ppo/humanoid-ppo.yaml

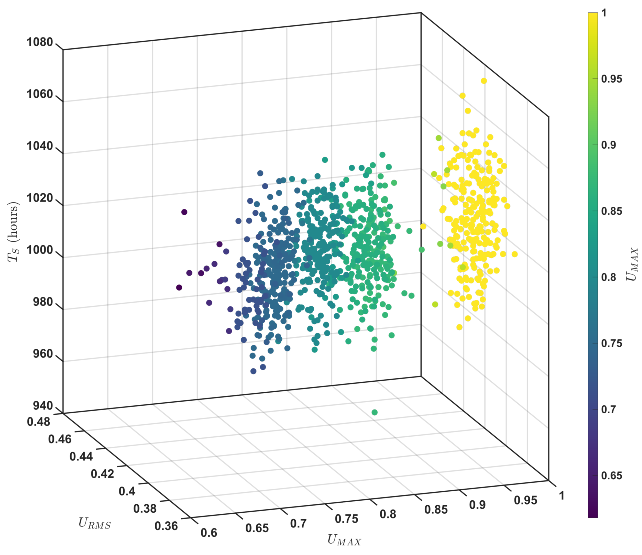


Figure 3: On week 40, the trained RL agent produced a best schedule of $T_S = 1{,}058$ hours, $U_{RMS} = 0.39$, $U_{MAX} = 0.82$. Relevant metrics of the solution space covered by 1,000 (trained) RL inference runs on week 40 of the SatNet dataset. Points are colored according to U_{MAX}; darker is better.

Table 3: **The RL formulation performs comparably to existing MILP baselines on the SatNet dataset.** Here, T_S represents the total hours scheduled, T_R represents the total hours requested, U_{RMS} represents the root mean square of the unsatisfied time fraction across all missions (described above), and U_{MAX} represents the unsatisfied fraction for the least satisfied mission.

Week	T_S / T_R (hours)	U_{RMS}	U_{MAX}	Number of satisfied requests	Training time (hours)
10	886 / 1192	0.28	0.71	204	4
20	1000 / 1406	0.27	0.81	223	18
30	1100 / 1464	0.28	0.85	229	13
40	1058 / 1737	0.39	0.82	216	6
50	879 / 1292	0.36	0.67	185	25

State Transition Heuristics

Two subroutines within the scheduling simulator require the specification of heuristics — the selection of a view peri-

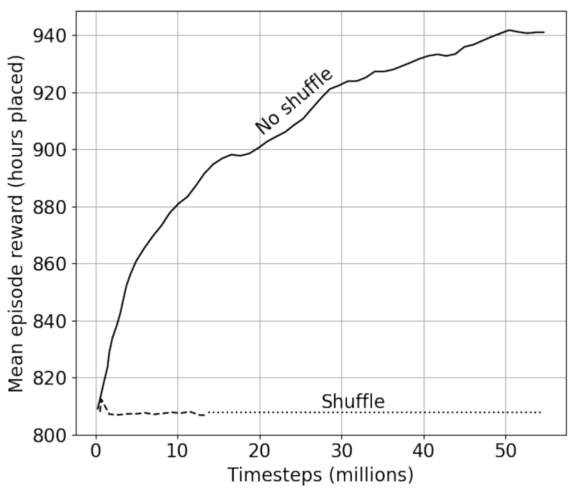


Figure 4: **Agents can memorize request sequences with shuffling turned off.** Comparison of agent's learning with and without shuffling.

ods $v_{i,j,k}$ from the set of valid view periods $\mathbb{V}_i$, and the alignment of the transmission time within $v_{i,j,k}$, should it be longer than the requested duration d_i.

VP Selection The selection of $v_{i,j,k}$ involves two heuristics, one to select the resource $a_{i,j}$, and another to select the k^{th} view period $v_{i,j,k}$ from $\mathcal{V}_{i,j}$. We implemented the following heuristics for the selection of $a_{i,j}$:

1. Choose the resource with the most available time
2. Choose the resource with the longest valid VP
3. Choose the resource with the highest *number* of valid VPs
4. Choose the resource with the least available time (i.e., the most utilized)
5. Choose a resource randomly

Once $a_{i,j}$ is selected, the simulator selects $v_{i,j,k}$ using one of the following heuristics:

1. Choose the longest VP on this resource
2. Choose the shortest VP on this resource
3. Randomly choose a VP from this resource

Shortening and Alignment As shown in Fig. 1, the simulator needs to allocate only the requested duration d_i in the event $v_{i,j,k}$ is longer than d_i. The user (or agent) can specify whether to align the resulting transmission time to the left, center, or right of $v_{i,j,k}$. In scheduling operations, mission planners have the option to specify analogous preferences — early, centered, or late.

In general, we found that the RL approach is insensitive to the choice of heuristic used in the simulator. All agents converged to similar levels of performance regardless of the combination of heuristics used, relying only on the state and reward signals from the environment to leverage its hidden transition mechanics and find optimal sequences of actions. Similarly, we also performed experiments using an RL formulation that exposed these heuristic choices to the agent, that is, the agent could decide not only the request index i, but provide guidance on the selection of j and k as well in $v_{i,j,k}$. Despite the increased time to convergence, the additional flexibility did not provide substantial performance benefits, indicating that 1) the idealized, single objective version of the scheduling problem can be solved just by memorizing/finding the optimal sequence of of actions for a given heuristic and/or 2) the state space's synoptic nature precludes the agent from identifying the best heuristic to choose.

Reward Functions

We extensively experimented with reward functions to maximise number of hours while providing fair allocation to all users. However, we empirically found that optimizing for number of hours allocated without any penalty helped achieve better results compared to other reward functions.

Conclusion and Future Work

In this paper, we presented a benchmark, SatNet, for satellite scheduling optimization based on historical data from the NASA Deep Space Network. We also presented the set of unique constraints that make this problem different from other combinatorial optimization problems. As discussed in the results section, the RL policies tend to overfit to the training dataset, however, we intend to focus on the generalization aspect as part of our future work. Although the yearly patterns have remained fairly consistent from historical data perspective, having generalizing models is beneficial from both product and technical standpoints. To close this gap, we have designed this benchmark for experimentation and invite contributions from the operations research and deep reinforcement learning community to improve upon the preliminary results obtained.

Reproducibility

All code, data, and experiments for this paper will be made available on the GitLab platform.

References

Balaji, B.; Bell-Masterson, J.; Bilgin, E.; Damianou, A.; Garcia, P. M.; Jain, A.; Luo, R.; Maggiar, A.; Narayanaswamy, B.; and Ye, C. 2019. Orl: Reinforcement learning benchmarks for online stochastic optimization problems. *arXiv preprint arXiv:1911.10641*.

Claudet, T.; Alimo, R.; Goh, E.; Johnston, M.; Madani, R.; and Wilson, B. Forthcoming. Δ-MILP: Deep Space Network Scheduling via Mixed-Integer Linear Programming. *IEEE Access*.

Clement, B. J.; and Johnston, M. D. 2005. The deep space network scheduling problem. In *Proceedings of the National Conference on Artificial Intelligence*, volume 3, 1514–1520.

Goh, E.; Venkataram, H. S.; Hoffmann, M.; Johnston, M. D.; and Wilson, B. 2021. Scheduling the NASA Deep Space Network with Deep Reinforcement Learning. In *2021 IEEE Aerospace Conference (50100)*, 1–10. IEEE.

Guillaume, A.; Lee, S.; Wang, Y. F.; Zheng, H.; Hovden, R.; Chau, S.; Tung, Y. W.; and Terrile, R. J. 2007. Deep space network scheduling using evolutionary computational methods. In *IEEE Aerospace Conference Proceedings*. ISBN 1424405254.

Johnston, M. D. 2008. An evolutionary algorithm approach to multi-objective scheduling of space network communications. *Intelligent Automation and Soft Computing*, 14(3): 367–376.

Johnston, M. D. 2020. Scheduling NASA's Deep Space Network: Priorities Preferences and Optimization. In *SPARK Workshop*.

Johnston, M. D.; Tran, D.; Arroyo, B.; Sorensen, S.; Tay, P.; Carruth, B.; Coffman, A.; and Wallace, M. 2014. Automated scheduling for NASA's deep space network. *AI Magazine*, 35(4): 7–25.

Kolen, A. W.; Rinnooy Kan, A.; and Trienekens, H. W. 1987. Vehicle routing with time windows. *Operations Research*, 35(2): 266–273.

Lee, K. D.; Cho, Y. H.; Lee, H. J.; and Jeong, H. 2002. Optimal scheduling for timeslot assignment in MF-TDMA broadband satellite communications. *IEEE Vehicular Technology Conference*, 56(3): 1560–1564.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 105400.

Nazari, M.; Oroojlooy, A.; Takáč, M.; and Snyder, L. V. 2018. Reinforcement learning for solving the vehicle routing problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 9861–9871.

Prins, C. 1994. An overview of scheduling problems arising in satellite communications. *Journal of the Operational Research Society*, 45(6): 611–623.

Sabol, A.; Alimo, R.; Kamangar, F.; and Madani, R. 2021. Deep Space Network Scheduling via Mixed-Integer Linear Programming. *IEEE Access*, 9: 39985–39994.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Tangpattanakul, P.; Jozefowiez, N.; and Lopez, P. 2015. A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite. *European Journal of Operational Research*, 245(2): 542–554.

Wong, Y. F.; Kegege, O.; Schaire, S. H.; Bussey, G.; Altunc, S.; Zhang, Y.; Chitra, P.; Kegege, O.; Schaire, S. H.; Bussey, G.; Altunc, S.; Zhang, Y.; and Chitra, P. 2016. An Optimum Space-to-Ground Communication Concept for CubeSat Platform Utilizing NASA Space Network and Near Earth Network. Technical report.

Xhafa, F.; and Ip, A. W. 2021. Optimisation problems and resolution methods in satellite scheduling and space-craft operation: a survey. *Enterprise Information Systems*, 15(8): 1022–1045.

Acknowledgments

This effort was supported by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The authors would like to thank the JPL Interplanetary Network Directorate and Deep Space Network team, as well as internal Quantum Scheduling Strategic Initiative team members Alex Guillaume, Shahrouz Alimo, and Thomas Claudet. U.S. Government sponsorship acknowledged.