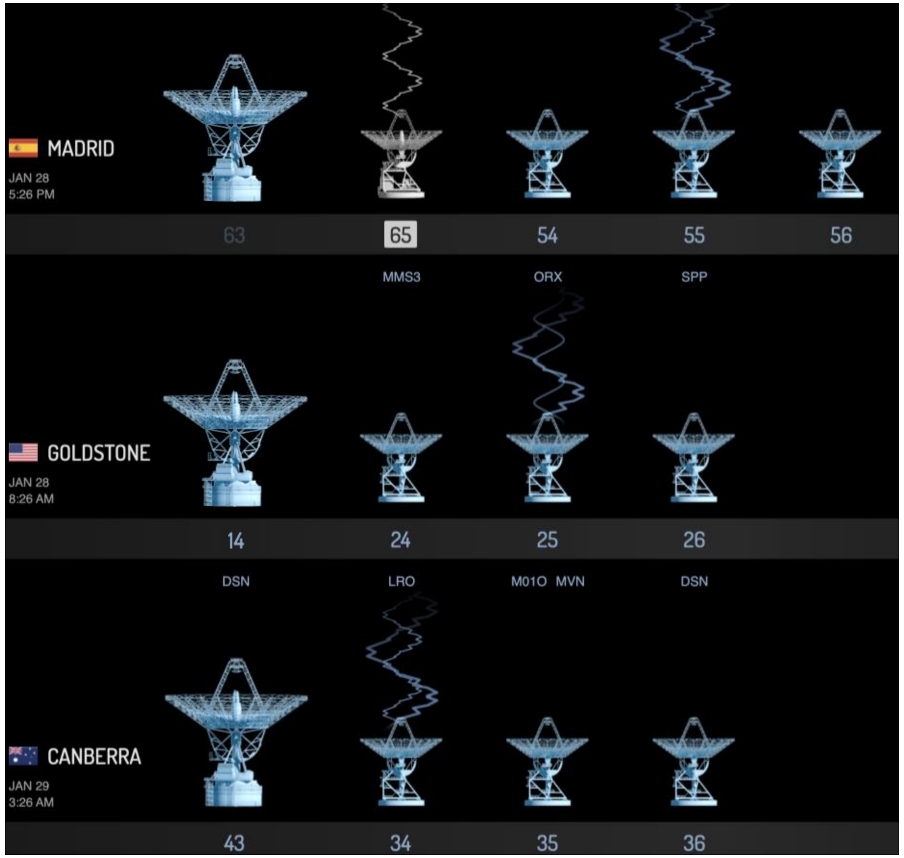
Jet Propulsion Laboratory
California Institute of Technology

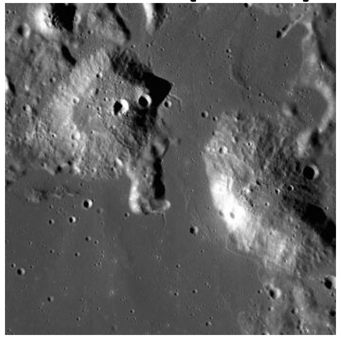
SatNet: A Benchmark for Satellite Scheduling Optimization

Edwin Goh, Hamsa Shwetha Venkataram, Bharathan Balaji, Mark D. Johnston, Brian Wilson

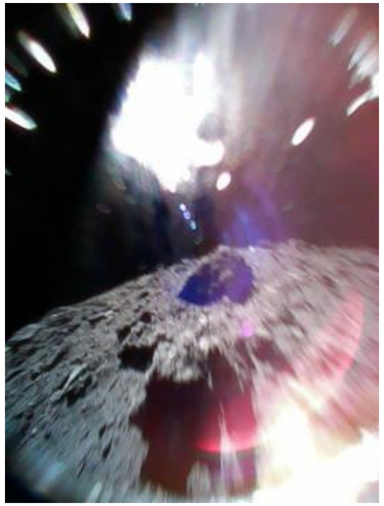
DSN Scheduling: Better Schedules in Less Time



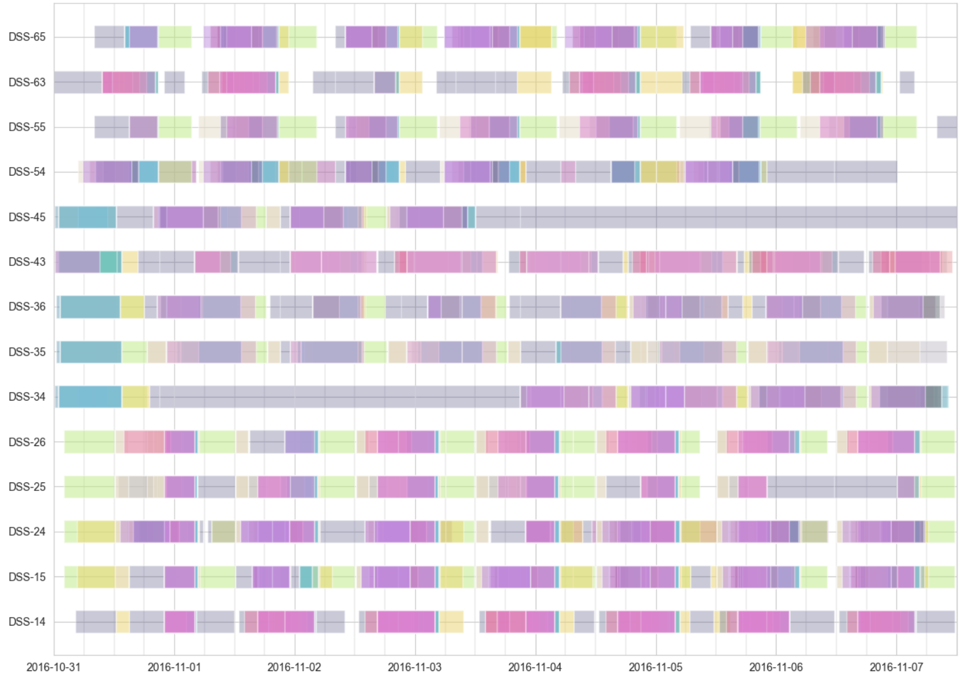
Lunar Reconnaissance Orbiter (2021)



Hayabusa 2 (2018)

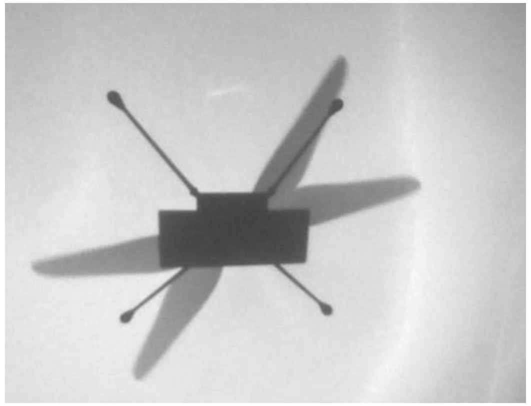


Example of overlapping visibility/view periods

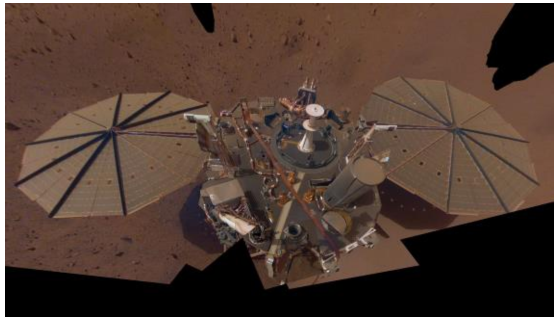


Voyager 1 (1979)

Ingenuity (2021)



Insight (2019)



New Horizons (2015)

The DSN Scheduling Problem

- DSN scheduling currently relies on peer-to-peer negotiation with **limited automation**
- Constraints: similar to other satellite scheduling problems
  - Visibility
  - Maintenance
  - Timing/preference
- Objective: Maximize satisfaction across all missions
- Potentially unique aspects:
  - Splitting of requests
  - Multi-antenna requests

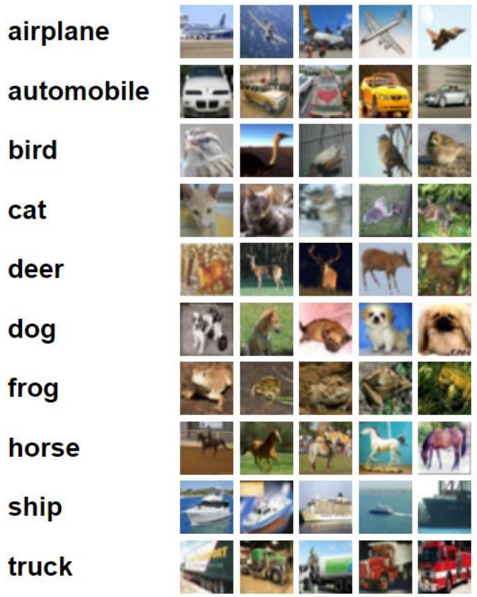
SatNet: A Benchmark to Compare Scheduling Techniques

- ML's rapid progress is supported by the development of benchmarks
  - Classification: MNIST, CIFAR, ImageNet
  - Segmentation: Cityscapes, PASCAL VOC, COCO
  - Language: Penn Tree Bank (PTB), SuperGLUE

- Satellite scheduling literature focuses on individual applications – difficult to compare & reproduce results

- SatNet uses **historical DSN data** from 2018
  - Mission requests
  - Spacecraft visibility
  - DSN antenna maintenance schedules

- SatNet includes an **initial reinforcement learning formulation**
  - Hopefully will kickstart other research efforts



SatNet Dataset Overview

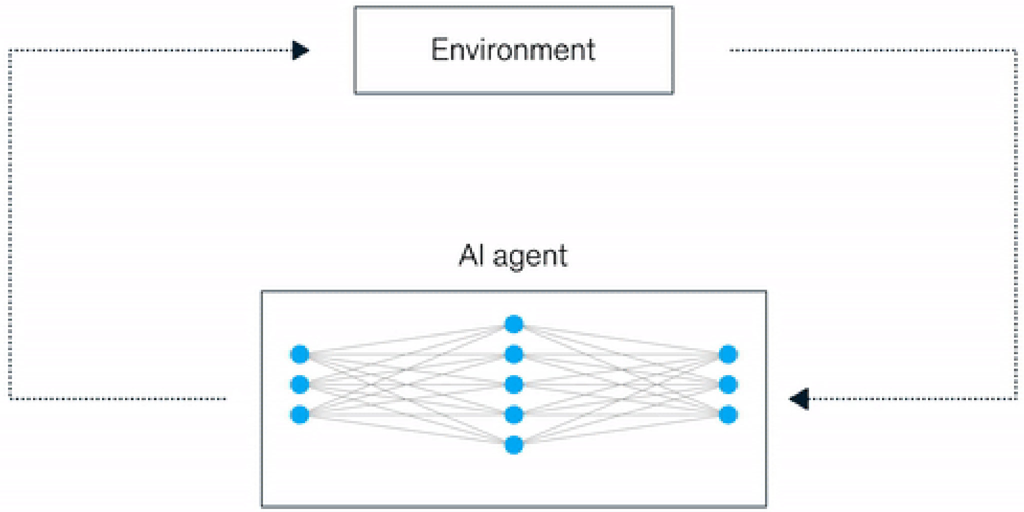
- Generated using **historical DSN data**
  - Mission requests
  - Spacecraft visibility
  - DSN antenna maintenance schedules
- Contains 5 weeks of data from 2018
  - Weeks 10, 20, 30, 40, 50
- Week 40 is particularly challenging
  - Total requested hours: 1,737

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


```json
{
    "subject": 521,
    "week": 10,
    "year": 2018,
    "duration": 1.0,
    "duration_min": 1.0,
    "track_id": "fc9bbb54-3-1",
    "setup_time": 60,
    "teardown_time": 15,
    "time_window_start": 1520286007,
    "time_window_end": 1520471551,
    "view_periods": {
      "DSS-34": [
        {
          "TRX ON": 1520286007,
          "TRX OFF": 1520318699
        },
        {
          "TRX ON": 1520398201,
          "TRX OFF": 1520410976
        }
      ],
      "DSS-36": [
        {
          "TRX ON": 1520286408,
          "TRX OFF": 1520318699
        },
        {
          "TRX ON": 1520391601,
          "TRX OFF": 1520410385
        }
      ]
    }
}
```


Example request from Week 10

Deep RL: Learning Scheduling Strategies vs. Finding Solutions



- Reinforcement learning (RL): a sub-field of machine learning where **agents** learn to make good decisions through trial-and-error in an **environment**

- Agent actually learns a **policy, π**
  - Deterministic: $\boldsymbol{a} = \boldsymbol{\pi}(\boldsymbol{s})$
  - Stochastic: $\boldsymbol{P}[\boldsymbol{\mathcal{A}} = \boldsymbol{a} | \boldsymbol{\mathcal{S}} = \boldsymbol{s}] = \boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s})$

Re-usable
- Train once, run many

Creative
- Stochastic policy
- Optimization under uncertainty
- Multiple candidate schedules

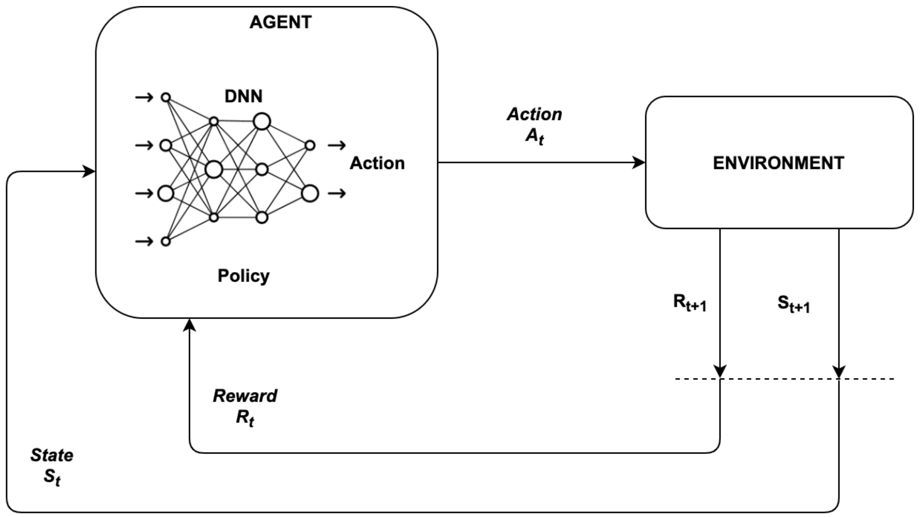
Flexible
- Based on simulation/digital twin
- Easily encode new DSN constraints, mission preferences, etc.

Scalable
- Parallel training on large clusters
- Parallel CPU-based inference

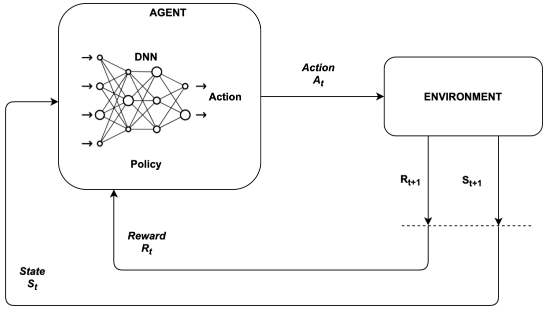
Dynamic
- Vibrant **open-source** community
- Incorporate **latest research** in DL and optimal control
  - Curriculum learning
  - Imitation learning
  - Self-supervised learning

Scheduling Simulator as the RL Environment

- Key elements in RL:
  - State
  - Action
  - Reward
  - **Environment**
- Environment takes actions, returns reward + next state
- Simulator manages dataset I/O, checks constraints, allocates valid requests



Two RL Formulations

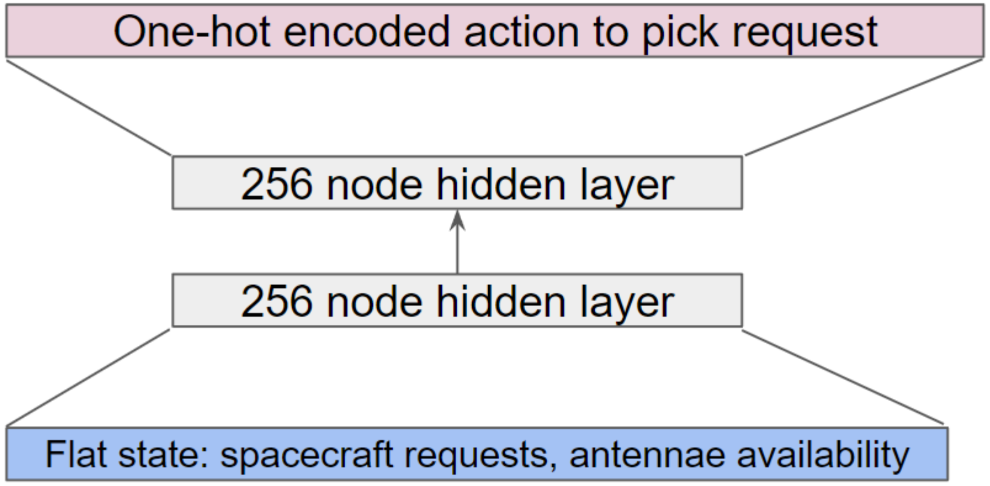


	"Batch" Scheduling	**"Online" Scheduling**
State	• Selected attributes from all requests • Available hours remaining on each antenna	• Requests stored in queue, provided to agent one at a time • All attributes for a single request – including valid view periods • Antenna availability • Uses **visual representation**
Action	• Pick request index • Invalid/unsatisfiable requests are masked out	• Pick antenna(s) • Pick time slot to schedule request • Decide whether request should be split
Reward	Total hours satisfied	
Environment	Simulator uses greedy heuristic to schedule the selected request	Simulator determines whether selected antenna(s) and time slot are valid for current request

State Representations

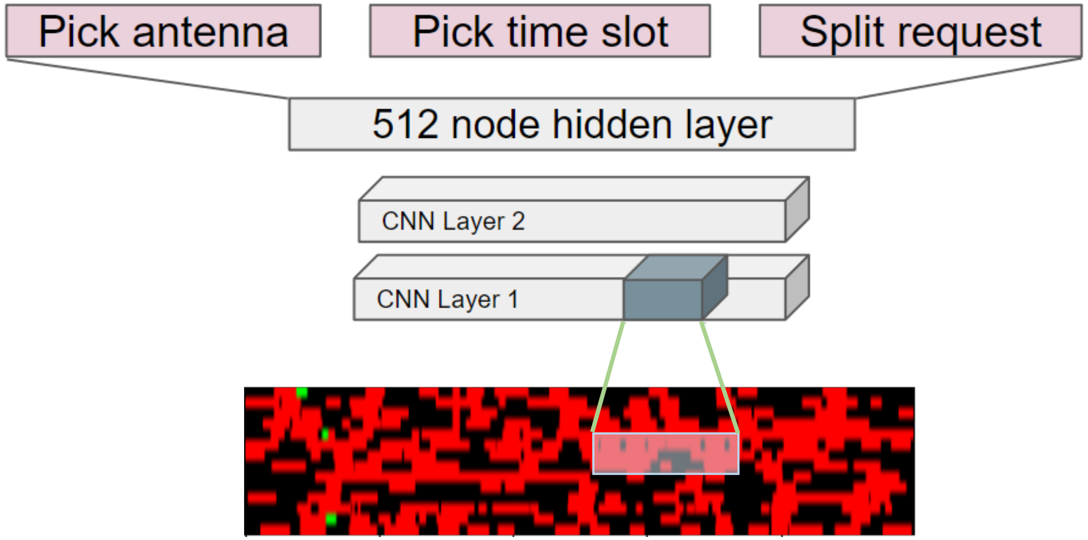
Batch Scheduling

- State: attributes from all requests, flattened array
- Simple discrete action space
- Fully connected network



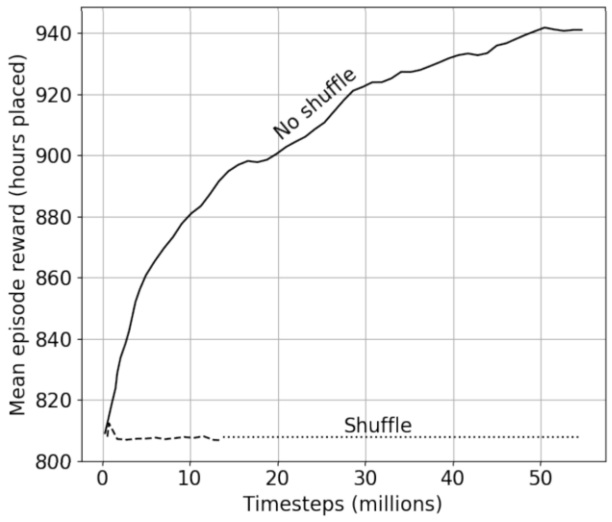
Online Scheduling

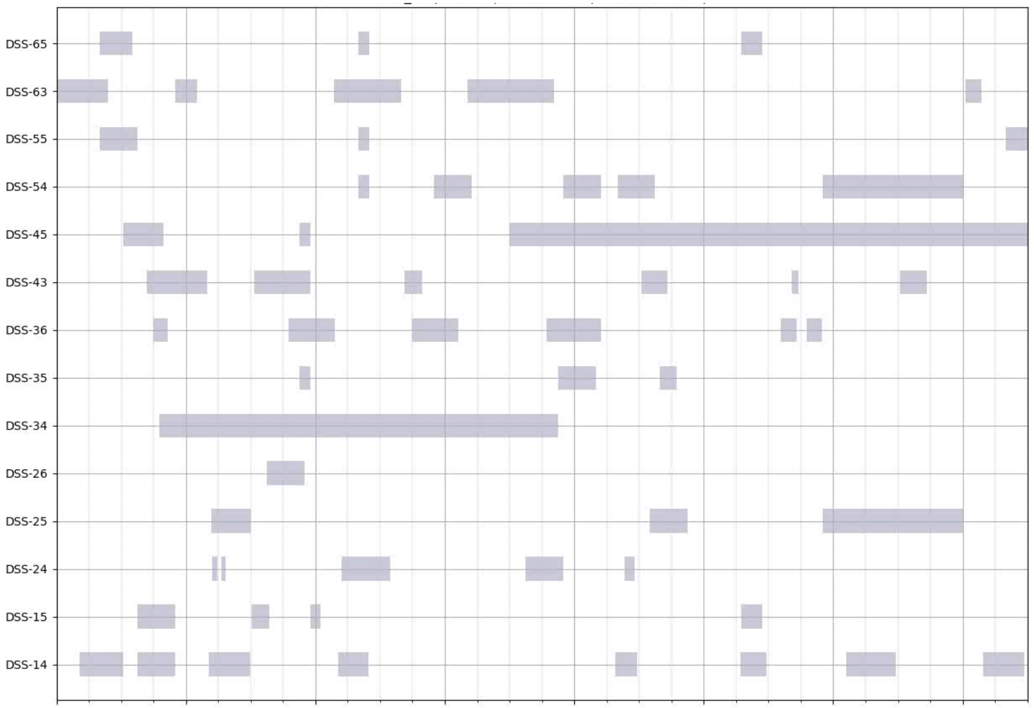
- State: Image with 2 channels
  - Channel 1: Antenna availability
  - Channel 2: Request attributes
- Multi-discrete action space
- Convolutional network



Batch Scheduling Agent Memorizes Request Order

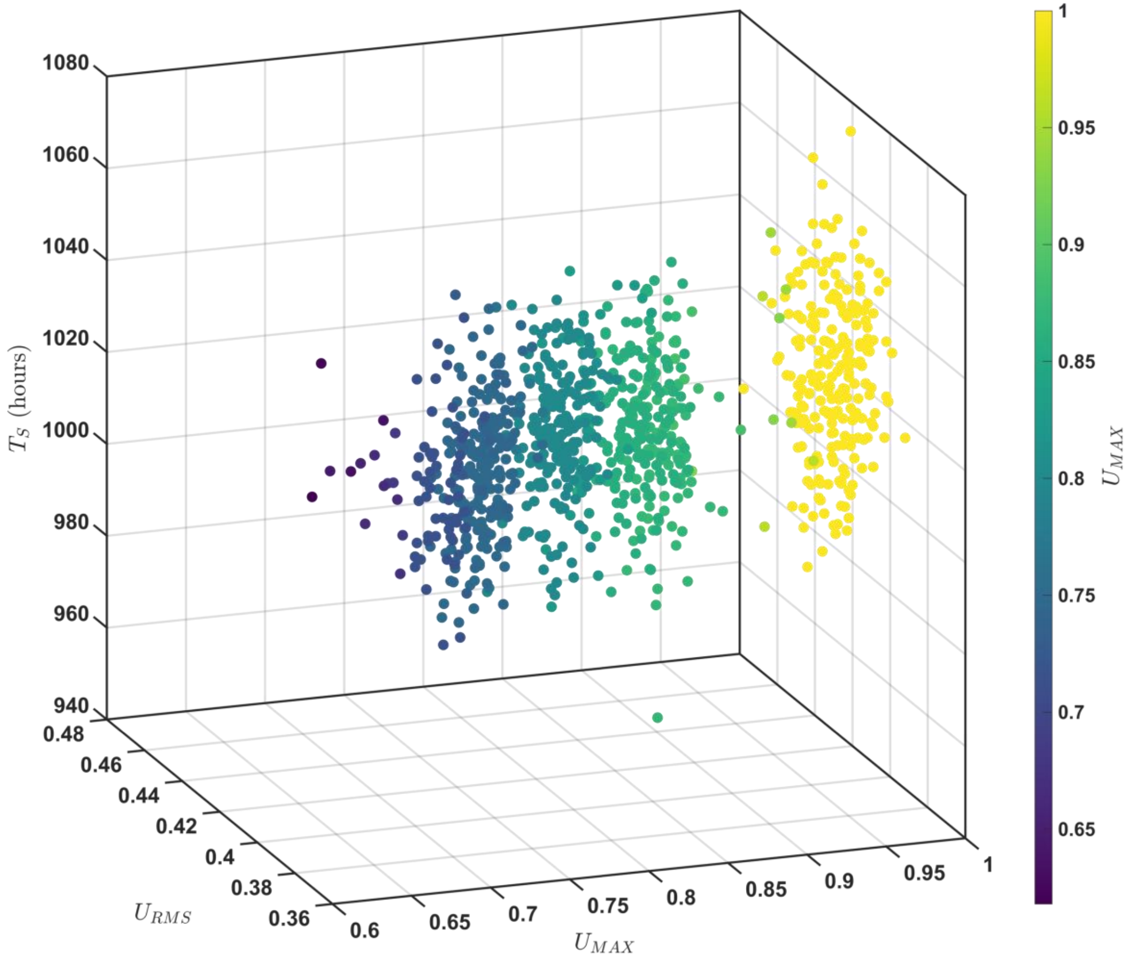
- Trained using Proximal Policy Optimization (PPO) on one week of data
- Without shuffling, agent memorizes order of requests to allocate
- With shuffling, agent performs close to random
  - i.e., fails to learn *generalizable* policy
- Preliminary results from online scheduling approach indicates that agents can learn on shuffled requests, but converge to lower number of hours





Candidate Schedules with Trained Stochastic Policy

- We have a stochastic policy, so each inference run can provide a different candidate schedule
  - Simulator ensures that all schedules are valid
- Run 1,000 iterations with trained agent (no shuffle)
  - Use same week as training
- Week 40, best out of 1,000 inference runs:
  - 1,058 hours allocated (out of 1,737 requested)
- Right figure shows 1,000 different solutions/schedules
  - T_s is the total scheduled hours
  - U_{RMS} and U_{max} are fairness metrics; lower is better



Conclusions

- We present the SatNet benchmark (dataset + RL implementation) for satellite scheduling problems
  - Available at: https://github.com/edwinytgoh/satnet
- We invite researchers to:
  - Apply various techniques (ML or otherwise) to SatNet and publish results
  - Contribute new metrics to facilitate comparisons on various objectives
  - Develop additional general-purpose scheduling benchmarks
- Initial RL results indicate that RL can memorize request order and find good solutions
  - This is akin to a (guided) search method; does not enable inference on other weeks/problem sets
- Future RL work:
  - Improve online scheduling environment with auto-regressive action space
  - Incorporate search-based algorithms, e.g., Monte Carlo Tree Search

Thank You!

Edwin Goh: edwin.y.goh@jpl.nasa.gov

Hamsa Shwetha Venkataram: hamsa.shwetha.venkataram@jpl.nasa.gov

Bharathan Balaji: bhabalaj@amazon.com

Brian Wilson: bdwilson@jpl.nasa.gov

Mark D. Johnston: mark.d.johnston@jpl.nasa.gov