A Deep Reinforcement Learning Application to Space-based Sensor Tasking for Space Situational Awareness

Thomas G. Roberts[*], **Peng Mun Siew**[†], **Daniel Jang**[‡]
Massachusetts Institute of Technology
Richard Linares[§]
Massachusetts Institute of Technology

ABSTRACT

To maintain a robust catalog of resident space objects (RSOs), space situational awareness (SSA) mission operators depend on ground- and space-based sensors to repeatedly detect, characterize, and track objects in orbit. Although some space sensors are capable of monitoring large swaths of the sky with wide fields of view (FOV), others—such as maneuverable optical telescopes, narrow-band and imaging radars, or satellite laser ranging systems—are restricted to relatively narrow FOVs and must slew at a finite rate from object to object as they observe them. Since there are many objects that a narrow FOV sensor could choose to observe within its field of regard (FOR), it must algorithmically create a schedule that dictates which direction to point and for how long: a combinatorial optimization problem known as the sensor tasking problem (Erwin et al. 2010). In this paper, we describe a specific application of a trained scheduler that was developed using deep reinforcement learning with the proximal policy optimization (PPO) algorithm (Jang et al. 2020) to a space-based narrow FOV sensor in low Earth orbit (LEO). The sensor's performance—both as a singular sensor acting alone, but also as a complement to a network of taskable, narrow FOV ground-based sensors—is compared to the greedy scheduler across several figures of merit, including the cumulative number of RSOs observed and the mean trace of the covariance matrix of all of the objects in the scenario. The results of several simulations are presented and discussed. Additionally, the results from a LEO SSA sensor in different orbits are evaluated and discussed, as well as various combinations of space-based sensors.

1. INTRODUCTION

As more resident space objects (RSOs) are added to the United States Space Command's (USSPACECOM) RSO catalog with the advent of proliferated satellite constellations and the deployment of more accurate sensors that can detect smaller objects, the need for exquisite space situational awareness (SSA)—the capabilities of detecting, cataloguing, and tracking RSOs—grows more critical for sustaining long-term space operations. There are currently more than 4,000 active satellites in low Earth orbit (LEO) [1] and it is estimated that by 2025 over 1,000 satellites could be launched each year. [2] The number of satellites will likely greatly outpace any increased capacity of SSA sensors, making efficient tasking of existing and future sensors—including both ground- and space-based narrow field of view (FOV) sensors—extremely valuable.

The SSA sensor tasking problem suffers from the *curse of dimensionality*, a challenging aspect of the problem wherein the complexity of the object-tracking problem grows exponentially as the number of targets and length of the observation window grows linearly. Scheduling agents trained using reinforcement learning methods have been shown in literature [3–5]. More recently the authors have developed and trained a scheduler for ground-based narrow FOV sensors to efficiently observe RSOs orbiting overhead using a deep reinforcement learning with the proximal policy optimization (PPO) algorithm and population-based training (PBT) [6, 7]. In this paper, the scheduler is adapted for a constantly moving space-based sensor. The adapted scheduler outperforms myopic policies—those in which only the benefits of a small number of observations into the future are considered—across several figures of merit, including RSO covariance and the number of unique RSOs observed during the study period.

[*]Ph.D. Candidate, Department of Aeronautics and Astronautics. E-mail: thomasgr@mit.edu
[†]Postdoctoral Associate, Department of Aeronautics and Astronautics. E-mail: siewpm@mit.edu
[‡]Ph.D. Candidate, Department of Aeronautics and Astronautics. E-mail: djang@mit.edu
[§]Boeing Assistant Professor, Department of Aeronautics and Astronautics. E-mail: linaresr@mit.edu

2. SPACE-BASED SENSORS

Although most of the sensors in the U.S. Space Surveillance Network (SSN) are ground-based, space-based sensors have contributed observations since the late 1990s. Over the past decade, several additional satellites have been launched to LEO and the geostationary (GEO) region to augment the SSN, including the U.S. military's Space Based Space Surveillance (SBSS) system and Geosynchronous Space Situational Awareness Program (GSSAP) satellites as well as the the Canadian military's Sapphire system and Near-Earth Orbit Surveillance Satellite (NEOSSat). While SBSS and Sapphire are taskable and gimbaled sensors that can be pointed, SensorSat is a body-fixed satellite for GEO surveillance in an equatorial LEO orbit. Much like taskable, limited FOV terrestrial sensors, gimbaled sensors in space will also benefit from optimized tasking, which will help maximize new sensors' utility to the SSN. Table 1 features a list of operational space-based sensors for SSA, including their operators, contractors, launch years, mass, orbit, and FOV, when available.

3. SSA ENVIRONMENT FORMULATION

OpenAI's Gym library was used to create a custom reinforcement learning environment. [19] The environment propagates the RSOs, computes the covariance, creates the observation data and computes the reward for the agent. RSO state propagation is done via SGP4 using the open source PyEphem library. [20] Each rollout of an episode is limited to a 60-minute finite horizon scenario. The number of RSO in the environment can be defined by the user. Covariance propagation and update is done using the Unscented Kalman Filter (UKF) formulation. The initial covariance for each RSO is randomized at the start of each episode according to a set distribution as used in [7]. State propagation is done using SGP4.

3.1 Sensor parameters

The sensor is assumed to have a 4 deg by 4 deg field of view (FOV) which can point anywhere in the 4 π-sr sphere except down to -14 deg elevation due to Earth limb exclusion zone. With 360 degrees in azimuth and 104 degrees in elevation to cover, this field of regard (FOR) can be fully covered by a 90×26 grid for a total of 2340 possible pointing directions. The slew rate of the pointing is assumed to be 2 deg/sec with a settle time of 4 sec. The geometry of the sensor's FOR and slew duration is shown in Fig. 1.

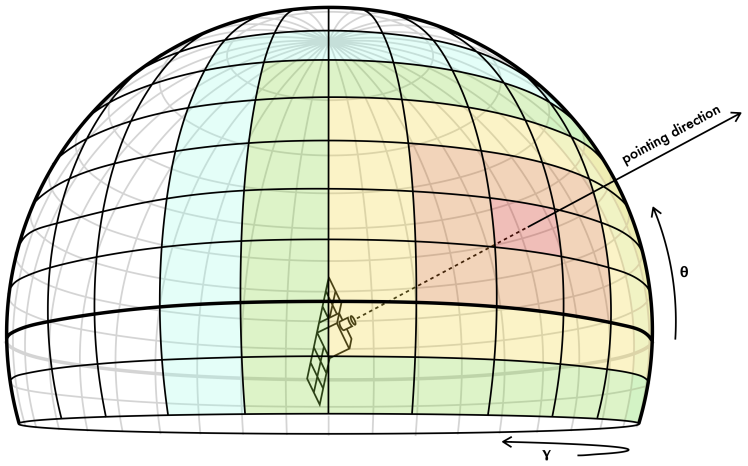


Fig. 1: Nominal field of regard for a low-altitude space-based SSA sensor.

3.2 RSO orbits

The observed RSO population is in near-GEO altitude (semi-major axis of 37000 to 45000 km) with low to moderate eccentricities (< 0.6) and no limitations to inclination, right ascension of ascending node (RAAN), argument of perigee, and anomaly. An example scenario of a SSA sensor in Sapphire orbit compared to the GEO-like RSOs is shown graphically in Fig. 2.

Table 1: Operational space-based sensors for SSA

Name	Operator	Contractor	Launch Year	Mass	Orbit	FOV (º)	Ref.
Space Based Visible (SBV) on the Midcourse Space Experiment (MSX)	U.S. Ballistic Missile Defense Office	MIT Lincoln Laboratory	1996	78 kg	898 km, near sun-synchronous	1.4×6.6	[8] [9]
Space Tracking and Surveillance System (STSS)	U.S. Missile Defense Agency	Northrop Grumman, Raytheon	2009	$\sim$1000 kg	1350 km, 58° inclination		[10]
Space-Based Space Surveillance (SBSS)	U.S. Missile Defense Agency	Northrop Grumman, Boeing, Ball Aerospace	2010	1031 kg	630 km, sun-synchronous	$\sim 2 \times \sim 4$	[11]
Sapphire	Canadian Department of National Defence	MacDonald, Dettwiler and Associates, Surrey Satellite Technology Ltd., COM DEV	2013	28.5 kg	786 km, sun-synchronous	1.4×1.4	[12]
Near-Earth Orbit Surveillance Satellite (*NEOSSat*)	Canadian Space Agency, Canadian Department of National Defence	Microsatellite Systems Canada Incorporated (MSCI)	2013	72 kg	785 km, 98° inclination	0.8×0.8	[13] [14]
Geosynchronous Space Situational Awareness Program (GSSAP)	U.S. Space Force	Orbital Sciences Corp.	2014				[15] [16]
Operationally Responsive Space 5 (*SensorSat*)	U.S. Space Force	MIT Lincoln Laboratory	2017	120 kg	600 km, equatorial		[17] [18]

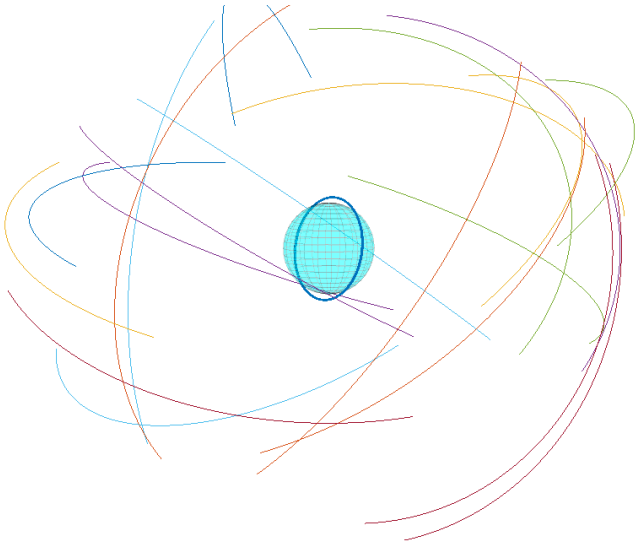


Fig. 2: Orbits of an example LEO SSA agent and 20 GEO-altitude RSOs propagated for 7 hours

3.3 Sensor action

The sensor's action is determined by evaluating the DRL agent that has been trained. The flowchart of this process for each step is shown in Fig. 3, and the training process is described in the following section. Once an action is chosen by the agent, the environment uses that information to calculate the action slew time. The RSOs' states and covariance are then propagated forward in time based on the action slew time. RSOs that are in the agent's FOV are identified and the environment performs the covariance update for those RSOs according to the sensor parameters described above.

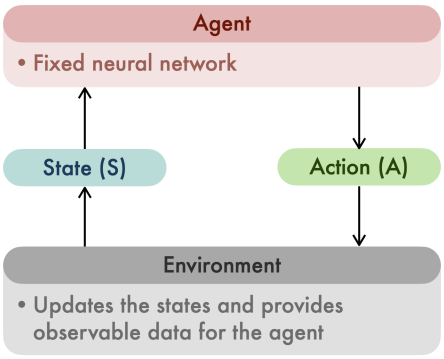


Fig. 3: Evaluation process for a single-sensor DRL agent

3.4 Reward

DRL agent training is inherently sensitive to the reward function used. Reward shaping can be used to guide some desired behavior from the trained agent. In the sensor tasking problem, there can be a multitude of objectives – for example, maximizing the number of RSOs observed may be preferable, or in other situations the maximum uncertainty of any RSO may need to be below certain threshold. For all of these performance metrics, the reward defined by the environment can be tweaked such that the trained agent performs well for the particular metric that the user emphasizes. In this paper our goal is to minimize the total uncertainty over all RSOs – that is, minimize the mean trace covariance across all RSOs at the end of our observation window.

Two reward functions are explored in this paper. The first reward function *rew1* is based on the time-discounted max trace reduction as shown in [6,7]. The second reward function *rew2* is simply the first reward without the time-discount factor. These rewards $R[t]$ at timestep k are shown in Equation 1 and 2 respectively.

$$R_{rew1}[k] = \frac{\arg\max_{a_k}\left[tr\left(P_{k|k-1}^{(a_k)}\right) - tr\left(P_{k|k}^{(a_k)}\right)\right]}{T[k] - T[k-1]} \tag{1}$$

$$R_{rew2}[k] = \arg\max_{a_k}\left[tr\left(P_{k|k-1}^{(a_k)}\right) - tr\left(P_{k|k}^{(a_k)}\right)\right] \tag{2}$$

where $tr\left(P_{k|k-1}^{(a_k)}\right)$ is the trace of the covariance for RSOs within the FOV selected by action a_k and $T[k]$ is the time at timestep k.

4. DEEP REINFORCEMENT LEARNING FORMULATION

The objective for the SSA system is to reduce the mean covariance of the RSO population during each episode. Two figures of merit are used to compare performance (1) the number of RSOs observed and (2) the mean covariance of all of the RSOs during the episode. Population Based Training (PBT) [21] was implemented using the Ray and Tune library. [22] Proximal Policy Optimization (PPO), a model-free DRL algorithm was used. [23] The hyperparameters for DRL training from [6, 7] were used.

4.1 Neural Network Formulations

A few neural network formulations were used for this paper based on the neural network architectures used in [7]. Table 2 has the relevant information on the neural network (NN) designs.

Table 2: Comparison of explored neural network architectures

Architecture	Input Size	Output Size	Layers
CNN v1	$90 \times 26 \times 11$	2340	Conv2d(32,8,4), Conv2d(64,4,2), Conv2d(64,3,1), FCL(1024), FCL(Output Size)
CNN v2	$90 \times 26 \times 11$	2340	Conv2d(16,8,4), Conv2d(32,4,2), Conv2d(32,3,1), FCL(700), FCL(Output Size)
CNN v3	$90 \times 26 \times 11$	2340	Conv2d(48,8,4), Conv2d(80,4,2), Conv2d(80,3,1), FCL(2048), FCL(Output Size)
CNN v4	$90 \times 26 \times 11$	2340	Conv2d(48,8,4), Conv2d(80,4,2), FCL(2048), FCL(Output Size)

4.2 Action Space

The pointing direction is the action chosen by the agent. In order to reduce the action space for the agent and for simplified modeling, the pointing direction was discretized into FOV grids. The FOV was chosen to be 4×4 degrees, a typical field of view of an optical sensor observing GEO satellites. A minimum elevation limit of -14 degrees is used, which yields an action space that is represented by a 90×26 grid for elevation positions and azimuth positions respectively. The -14 degree elevation limit means that for a sensor in circular orbit, it is able to point 14 degrees towards Earth from its velocity vector.

4.3 Observation Space

Each of the azimuth-elevation grid has a number of data associated with it to describe the state of the environment for that region. These observables are calculated by the environment and passed onto the DRL agent. The choice of which observable data is calculated and passed onto the DRL agent is flexible. However, less meaningful data will simply act as noise to the training process and lead to slower training convergence and evaluation and possibly a failed training cycle.

The 11 observation data for each grid for the agent is shown in Table 3. These values are calculated at each step of the episode for each az-el grid by the environment and is passed onto the agent. The current pointing direction is a boolean value, where it is 1 if the corresponding grid is the current pointing direction and 0 if not. Though the observation grid is reoriented every time the sensor slews such that the azimuth is centered in the grid, the elevation grid is free to be on any row as shown in Fig. 4.

The observation grid is partitioned into 23 regions based on the approximate action slew time as shown in Fig. 4, where each color corresponds to a different approximate action slew duration. A different propagation time is used to populate the data for each of these regions to better capture the true expected relative location of the RSO due to the high relative velocity between the RSO and the space-based observer.

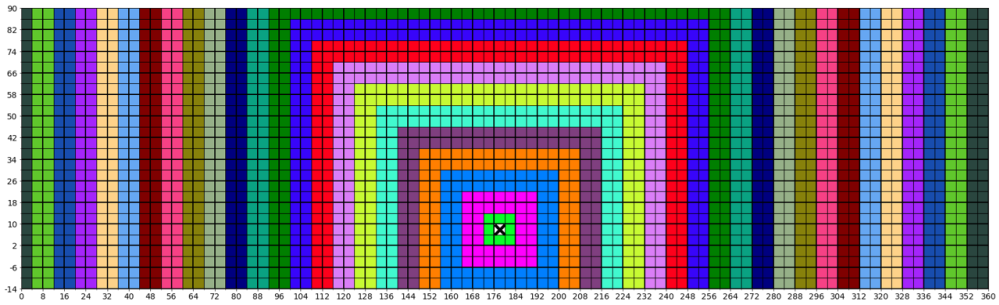


Fig. 4: Field of regard and propagation gradient.

Table 3: Observation information for each grid

Layer	Data (per observation grid)
1	Number of RSOs
2	Elevation fraction location of RSO
3	Azimuth fraction location of RSO
4	Range of RSO
5	Elevation velocity of RSO
6	Azimuth velocity of RSO
7	Range velocity of RSO
8	Max trace covariance of RSOs
9	Sum of RSOs trace covariance
10	Mean of RSOs trace covariance
11	Current pointing direction

4.4 Training

A flowchart describing the DRL training process is shown in Fig. 5. Training and evaluation were done on the MIT Lincoln Laboratory Supercomputing Center (LLSC). [24] The hyperparameters used for training were identical to that of [7].

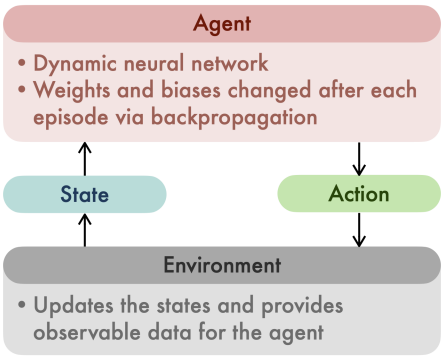


Fig. 5: Training process flowchart for a sensor-tasking DRL agent

The evolution of the mean reward for agent CNN v1 through training is shown in Figure 6. The means reward 1, rew1_100 term, was the mean reward obtained from agent CNN v1 with 100 RSOs in the training environment. The real-time training duration was between 67 and 77 sec per iteration for the environment with 100 satellites and 36 sec per iteration for 400 satellites.

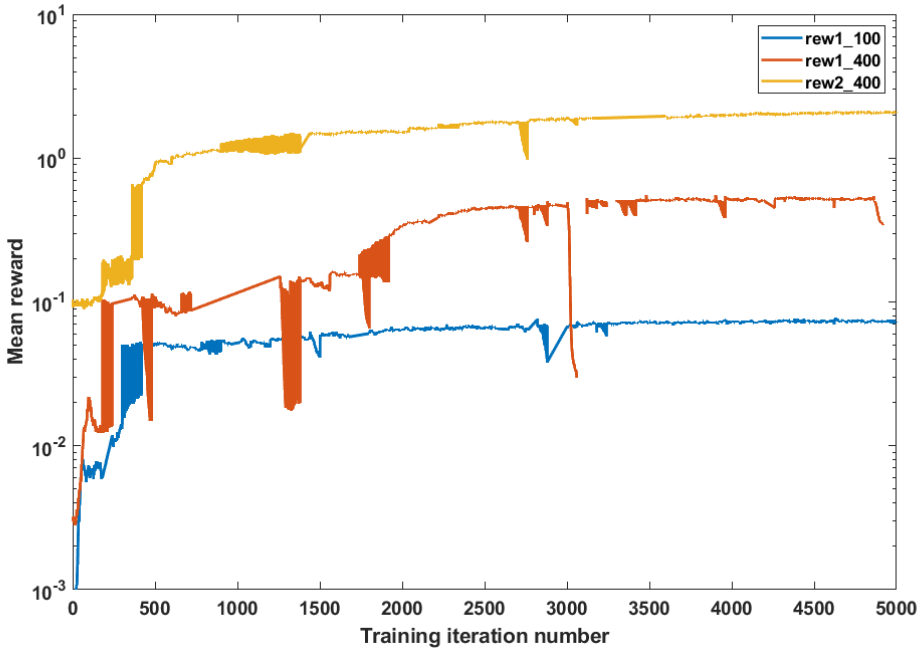


Fig. 6: Training statistics for three reward functions

4.5 Baseline myopic methods

Two baseline myopic algorithms are used as described in [7] but adapted for an on-orbit platform. First is a greedy algorithm where the sensor points to a RSO within the valid FOR with the highest covariance no matter the slew duration. An "advanced greedy" algorithm is developed where the covariance of each RSO within the FOR is normalized to the slew duration for the sensor to point to it. As there is a limited window of time to observe RSOs and lower the average covariance, considering the slew duration to observe is an important aspect. Of the two advanced greedy methods, one uses a time discount factor of $D = 5$ whereas another uses $D = 10$. The action policy of the "advanced

greedy" algorithm is given by 3.

$$a^* = \arg\max_a \left(tr[P^x_{i,k}] \cdot \delta t_i^{-1/D} \right) \tag{3}$$

where $tr[P^x_{i,k}]$ is the trace of a posteriori state covariance for RSO i at the current time step k, δt_i is the action slew time to move from the current pointing direction to observe RSO i, and D is a time discount factor.

4.6 Multi-agent formulation

There are many methods of implementing a multi-agent formulation. A scheduling agent could be created to oversee multiple sensors. The action-space and observation-space for such a scheduler would grow rapidly with each sensor and the number of sensors as well as the placement of each sensor may require a distinct trained agent suited for that setup.

Another method of implementing a multi-agent formulation would be to use multiple single-agents in parallel, with the environment tasked with querying the agents at the appropriate times. The environment will propagate all orbits until the next agent is available at which point the agent will be queried for its selected action.

This method of using multiple instances of the same trained agent in the same environment is explored in this paper. A scheduler is in charge of keeping track of which agent has completed their action. The flowchart of the process is shown in Fig. 7. This method allows the user during evaluation-time to place any number of agents as needed for a particular scenario to be executed. For evaluation we use an environment with 3 distributed agents.

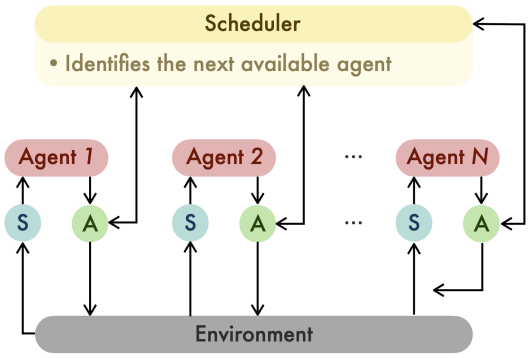


Fig. 7: Evaluation process for a multi-sensor approach

5. RESULTS AND DISCUSSION

We evaluate the trained agents and the myopic agents in several different environments. The observer sensor is placed in a polar orbit (Sapphire), inclined orbit (STSS) and equatorial orbit (SensorSat) for 100 GEO RSOs and 400 GEO RSOs. For each of these environments, a statistical comparison is done via Monte-Carlo runs with randomized RSO locations within the specified orbit bounds.

A typical pointing history for an episode is shown in Fig. 8. This shows a couple of interesting characteristics. The saw-tooth pattern of azimuth shows that the agent is rotating around itself to point in a systematic manner. Some runs show a clockwise rotation while others show a counterclockwise rotation, so no preferential direction is shown for the trained agents. However, this pattern does show that the agents learn some efficient method to search over the FOR, as jumping around in azimuth would expend too much duration to slewing.

Another interesting characteristic is the fact that most of the observation happens near 0 degree elevation, which is the tangential plane to the space-based sensor's orbit. This may be so such that the most number of RSOs may be gathered within the FOV, which would maximize the reward. For each of the observations, the agents chose a grid that often had 2-3 RSOs.

5.1 Training results using different reward functions

DRL agents were trained using two different reward functions *rew1* and *rew2* as described in Section 3.4. These agents were trained in an environment with 400 random RSOs in GEO-like orbits. After training reached convergence, these

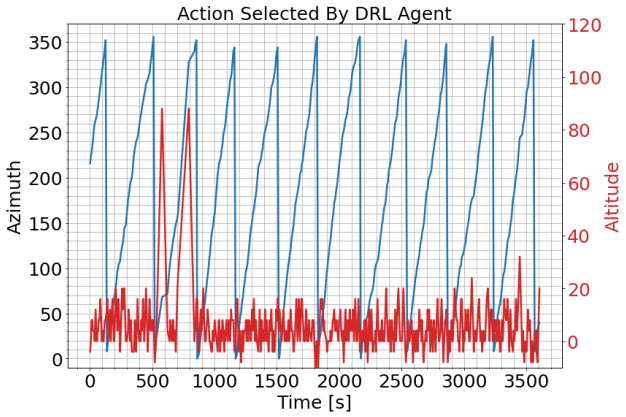
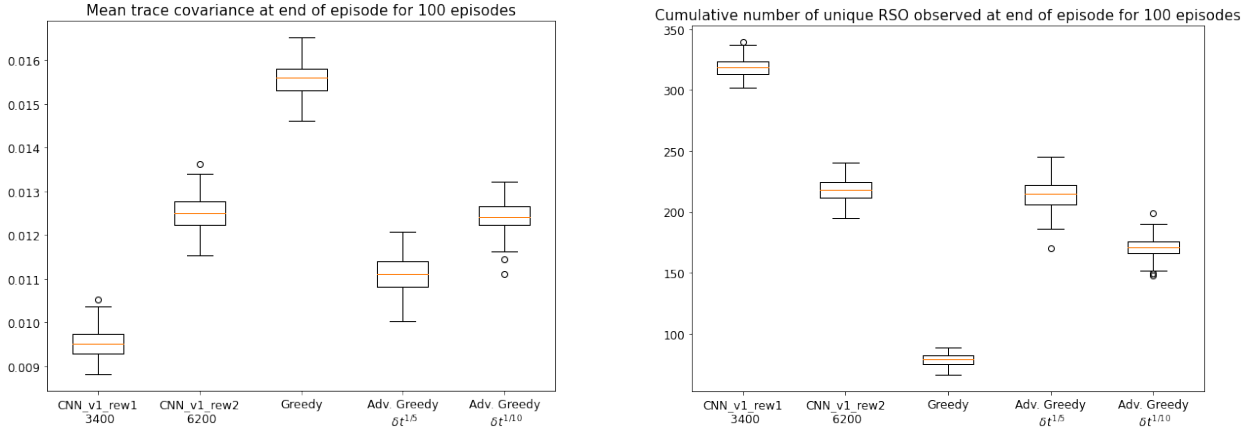


Fig. 8: An example of pointing directions chosen by the DRL agent in an example episode

were compared to the three myopic methods. 100 Monte Carlo simulations were done with the trained agents, and their final results showing the final mean covariance as well as the total number of unique RSOs observed are plotted in Fig. 9.



(a) Final mean covariance (lower is better)

(b) Total number of RSOs visited (higher is better)

Fig. 9: Comparison of results for agents trained with various reward functions. Note: *CNN_v1_rew1 3400* represents a CNN based DRL agent v1 trained for 3400 training iterations using *rew1*

As expected, the reward functions can have a large impact on the performance of the trained agents. *rew1* performs the best out of all other DRL and myopic methods with the lowest mean covariance and highest number of RSOs observed. The time-discounted factor in *rew1* seems to outperform the non-time-discounted reward function.

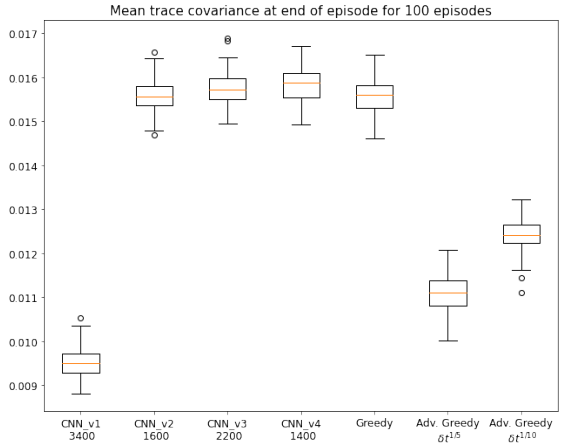
5.2 Training results using different neural network architectures

Several neural network formulations were explored as described in Section 4.1. Comparison between these four formulations trained with 400 RSOs is shown in Fig. 10. CNN v1 clearly performs the best out of all of these including the myopic methods. Note that this agent was trained the longest at 3400 iterations and has a NN architecture that is within the scope of the other NNs.

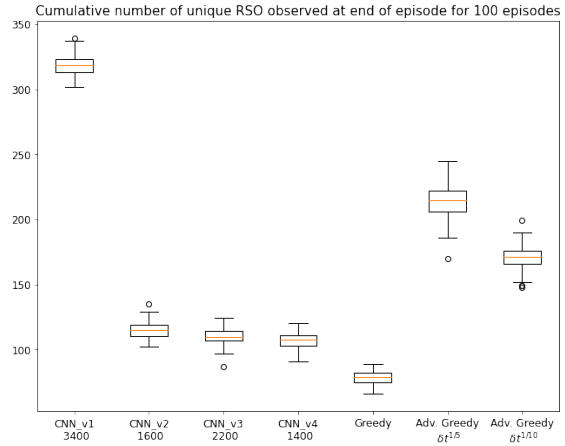
5.3 Training results using different number of RSOs

Next, we do a comparison between agents that were trained in an environment with 400 random RSOs in GEO-like orbits to those that were trained with 100 random RSOs. The results are shown in Fig. 11.

It is interesting to note that the agents trained in an environment with 400 RSOs always perform better than the agents trained in 100 RSOs. This is despite the fact that the agents trained with 100 RSOs is trained with more episodes.



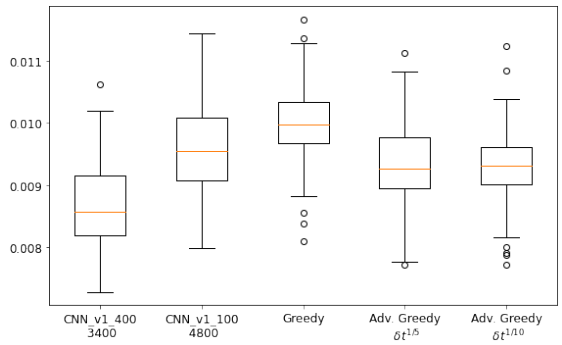
(a) Final mean covariance (lower is better)



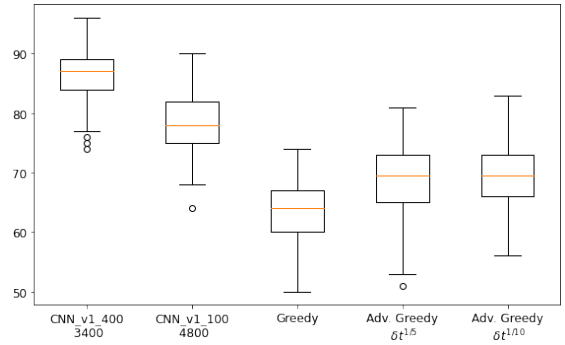
(b) Total number of RSOs visited (higher is better)

Fig. 10: Comparison of results for agents trained with various RSO numbers. Note: *CNN_v1 3400* represents a CNN based DRL agent v1 trained for 3400 training iterations using *rew1*

This may be due to the fact that with fewer RSOs the agent has such a sparse set of possible RSOs to choose from. In the environment with 400 RSOs there are more options to choose from, leading efficient learning of a more optimal policy. Comparison to even higher number of RSOs would give more insight in to this phenomena.



(a) Final mean covariance (lower is better)



(b) Total number of RSOs visited (higher is better)

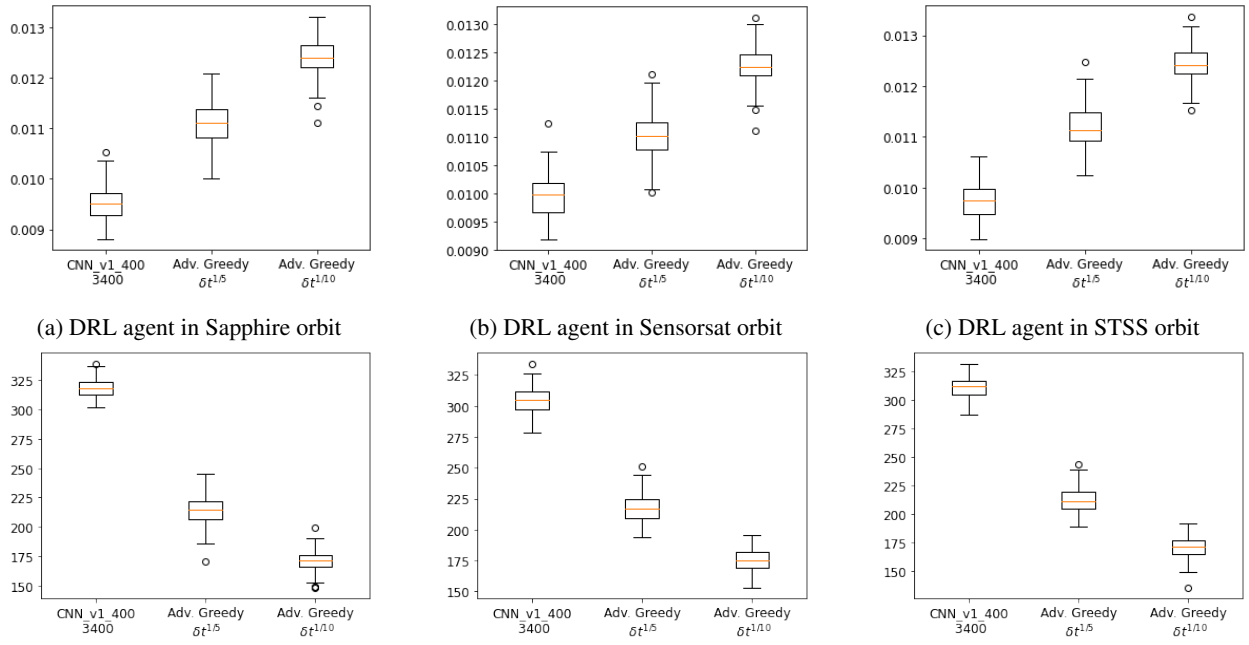
Fig. 11: Comparison of results for agents trained with various RSO numbers. Note: *CNN_v1 3400* represents a CNN based DRL agent v1 trained for 3400 training iterations using *rew1*

5.4 Robustness to changes in orbital plane

For any trained DRL agent, it is important to note its performance in an environment that is different from the one in which it was trained. The limits of its flexibility and robustness will allow for one trained agent to be used in multiple different scenarios, saving time for more trained agents specific to a particular environment.

For the space-based sensor tasking agent, it may be possible that an agent that was trained in a particular orbit may not perform as well in another orbit. To explore this problem, one DRL agent was trained in the Sapphire-like orbit (786 km altitude, 98.6 deg inclination) with 400 GEO-altitude RSOs and placed in other LEO SSA orbits such as STSS (1350 km, 58 deg) and SensorSat (600 km, 0 deg). The results for CNN v1 is shown. 100 Monte Carlo runs were done for each of these three cases and finals results compared, which are shown in Fig. 10.

Though the DRL agent was trained in a sun-synchronous orbit and performed well against the myopic methods, it also performed well in two different inclined orbits. This shows robustness of the agent to be used in different orbits. This flexibility allows one such agent to be extended to be used in other orbits, eliminating the need to train an agent that is specific to each orbital regime. This may be due to the fact that the observation data provided by the environment to



(a) DRL agent in Sapphire orbit (b) DRL agent in Sensorsat orbit (c) DRL agent in STSS orbit

(d) DRL agent in Sapphire orbit (e) DRL agent in Sensorsat orbit (f) DRL agent in STSS orbit

Fig. 12: Mean trace covariance (a - c) and number of unique RSOs observed (d - f) at the end of the episode using an DRL agent trained in Sapphire orbit evaluated in different orbits (lower is better). Note: *CNN_v1_400 3400* represents a CNN based DRL agent v1 trained for 3400 training iterations using *rew1* with 400 RSOs

the agent during training is consistent and independent of the orbit in which the sensor resides. For example, the same coordinate space (az-el) is used for the observation grid with the center column always corresponding to the current pointing direction. This shows that clever ways to formulate the problem may allow a more robust agent for varying use cases. In future work, the limits of this robustness can be explored, with the agent placed in eccentric or much higher altitudes. Training using GEO RSOs extended to LEO RSOs could also be investigated.

5.5 Multi-agent Environments

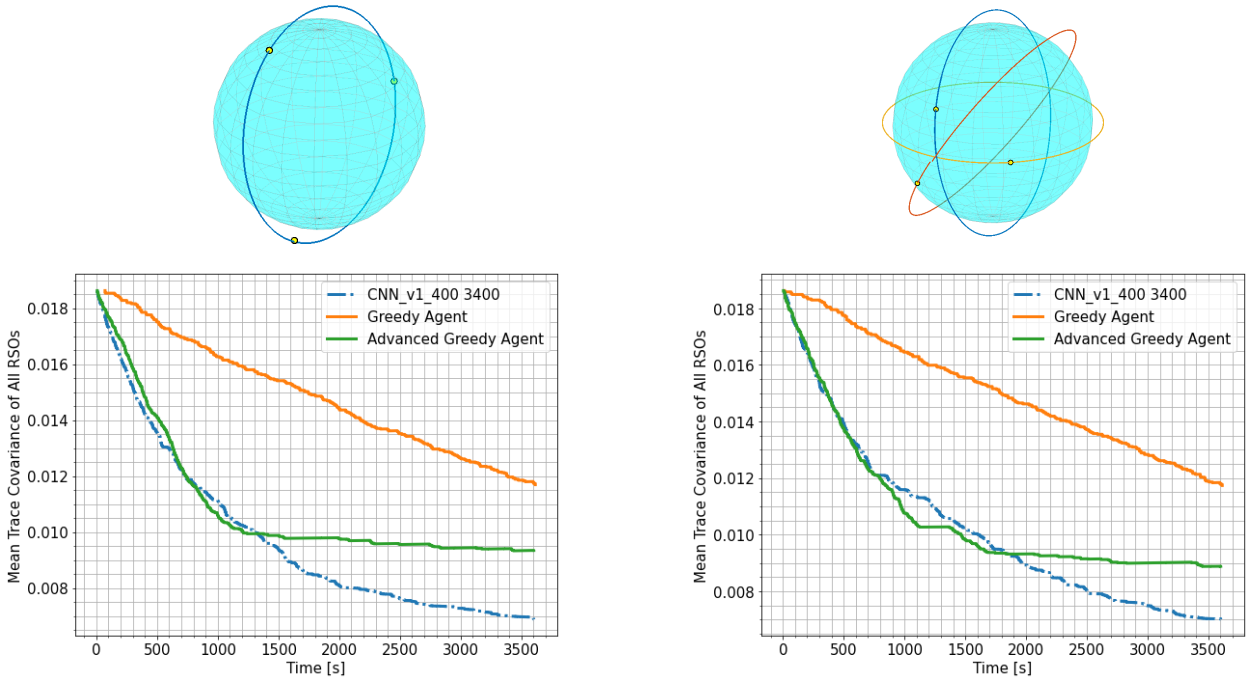
A couple of multi-agent formulations were evaluated in this work, both with three space-based SSA sensors. For the first case, three satellites are equally spaced within the Sapphire orbit. Each of these satellites are controlled by a DRL agent trained in the Sapphire orbit with 400 RSOs. The second formulation uses three different satellite orbits with one in the Sapphire orbit, another in the STSS orbit and another in the SensorSat orbit.

One episode was run for each of these cases, and the resulting mean trace covariance change during the episode is shown in Fig. 13. As shown in these figures, CNN v1 outperforms the myopic methods in both scenarios, ending the episode with a lower mean trace covariance. This multi-agent formulation is on-going work and will require further investigation, though it is clearly seen that this method shows promise.

6. CONCLUSION

Previous work have shown that terrestrial sensor tasking for SSA was possible using DRL. Space-based SSA sensors are being used today, and their relatively high performance and high cost means that efficient tasking of these tasked sensors is an important for SSA networks. In this paper we show that the sensor tasking problem for a space-based SSA sensor can be trained using DRL and perform well compared to myopic methods as well. We also show that a set of space-based sensors can also perform well using the single-sensor agent in a parallel manner, allowing for flexibility in the multi-agent formulation.

Future work for this work may include exploration of recurrent neural networks (RNN) such as long short-term memory (LSTM) for this problem, as the problem is highly time-series dependent. For the multi-agent problem, training



(a) 3 Sapphire satellites in one orbital plane

(b) One Sapphire, one Sensorsat and one STSS satellite

Fig. 13: Comparison of two different multi-agent formulations. Note: *CNN_v1_400 3400* represents a CNN based DRL agent v1 trained for 3400 training iterations using *rew1* with 400 RSOs

a multi-agent scheduler as opposed to the parallel single-agent method that we've used in this paper. Higher fidelity sensor modeling can be used to differentiate between the optical, radar and satellite laser ranging sensors. A combination of space-based and ground based sensors can also be used for a higher-fidelity simulation of the SSA architectures used today.

ACKNOWLEDGEMENT

The authors wish to acknowledge support of this work by the Air Force's Office of Scientific Research under Contract FA9550-18-1-0115 and the Aerospace Corporation's University Partnership Program. The authors would like to thank the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing (HPC, database, consultation) resources that have contributed to the research results reported within this paper.

7. REFERENCES

[1] Union of Concerned Scientists. UCS Satellite Database, 2021.

[2] T. Ryan-Mosley, E. Winick, and K. Kakaes. Boeing and DirecTV are scrambling to move a satellite before it explode, 2020.

[3] Bryan D. Little and Carolin E. Frueh. Space situational awareness sensor tasking: Comparison of machine learning with classical optimization methods. *Journal of Guidance, Control, and Dynamics*, 43(2):262–273, 2020.

[4] R. Linares and R. Furfaro. Dynamic sensor tasking for space situational awareness via reinforcement learning. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, 2016.

[5] R. Linares and R. Furfaro. An autonomous sensor tasking approach for large scale space object cataloging. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, 2017.

[6] D. Jang, P.M. Siew, G. Gondelach, and R. Linares. Space Situational Awareness Tasking For Narrow Field Of View Sensors: A Deep Reinforcement Learning Approach. In *71st International Astronautical Congress*. International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law, 2020.
[7] P.M. Siew, D. Jang, and R. Linares. Sensor Tasking for Space Situational Awareness Using Deep Reinforcement Learning. In *AIAA/AAS Astrodynamics Specialist Conference*, Big Sky, MT, 2021.
[8] G.H. Stokes, C. von Braun, R. Sridharan, D. Harrison, and J. Sharma. The space-based visible program. *Lincoln Laboratory Journal*, 11(2):205–38, 1985.
[9] European Space Agency. MSX (Midcourse Space Experiment).
[10] Gunter's Space Page. STSS 1, 2, 2021.
[11] U.S. Space Force. Space Based Space Surveillance, 2017.
[12] P. Maskell and L. Oram. Sapphire: Canada's Answer to Space-Based Surveillance of Orbital Objects. In *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, HI, 2008.
[13] Scott, R. and Thorsteinson, S. Key Findings from the NEOSSat Space-Based SSA Microsatellite Mission. Technical report, Defence Research and Development Canada, Maui, HI, 9 2018.
[14] European Space Agency. NEOSSat (Near-Earth Object Surveillance Satellite).
[15] U.S. Air Force. Geosynchronous Space Situational Awareness Program, 2017.
[16] Gunter's Space Page. GSSAP 1, 2, 3, 4, 5, 6 (Hornet 1, 2, 3, 4, 5, 6), 2021.
[17] MIT Lincoln Laboratory. ORS-5 SensorSat, 2021.
[18] Gunter's Space Page. ORS 5 (SensorSat), 2021.
[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zarema. OpenAI Gym, 2016.
[20] Rhodes, B.C. PyEphem: Astronomical Ephemeris for Python, 2011.
[21] M. Jaderberg, V. Dalibard, S. Osindero, W.M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu. Population Based Training of Neural Networks, 2017.
[22] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M.I. Jordan, and I. Stoica. Ray: A Distributed Framework for Emerging AI Applications. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad, CA, 10 2018. USENIX.
[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, 2017.
[24] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas. Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis. In *Proceedings of the IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, 7 2018. Institute of Electrical and Electronics Engineers.