12th International Workshop on Satellite and Constellations Formation Flying, Kaohsiung, Taiwan, December 2-4 2024.

Scalable Autonomous Decentralized Constellation Tasking on Asynchronous Semi-Markov Decision Processes

Mark A. Stephenson* **and Hanspeter Schaub†**

Spacecraft and constellation scheduling is a challenging problem, especially when considering uncertain environments and many agents. Traditional methods often rely on expensive ground-based planners to generate task schedules, which may not be possible in highly dynamic environments. This paper presents a scalable, decentralized, autonomous approach to constellation tasking by formulating the problem as an asynchronous decentralized partially observable semi-Markov Decision Process (Dec-POsMDP) and using deep reinforcement learning to train a single per-agent policy for all agents to execute locally. In particular, wildfire observation is considered as a motivating science objective, in which satellites in a constellation must opportunistically image fires of unknown importance. This application highlights two benefits of using reinforcement learning for spacecraft tasking that emerge from the decentralized autonomous approach: 1) closed-loop responsiveness to opportunistic task lists; and 2) scalability to constellations of differing sizes. The benefits of fine-tuning the policy for a particular constellation architecture in a multiagent environment are demonstrated through a comparison of policies over different constellations, and the generalizability of policies to larger constellations is investigated.

Keywords: reinforcement learning, autonomy, satellite constellations, spacecraft tasking, wildfire monitoring

1. Introduction

Planning and scheduling for satellites and constellations is challenging due to the complexity associated with generating and uplinking task schedules. This is further compounded when considering highly nonstationary environments, such as those with many opportunistic events to exploit, which can lead to a cascading need for on-the-fly replanning that may not be feasible with onboard resources [1]. Per-agent, closed-loop distributed autonomy is an attractive alternative to global open-loop planners for opportunistic science missions [2], presenting new challenges but enabling otherwise impossible mission architectures. In this work, wildfires are opportunistic and quickly evolving events that are valuable to responsively monitor for both scientific and disaster response purposes; this paper will present a scalable, decentralized, autonomous approach to wildfire monitoring with a constellation of satellites.

Traditional methods for satellite scheduling tend to rely on global or hierarchical ground-based planners to generate constellation-wide task sequences and upload them when satellites are accessible [1, 3]. References [4–7] and others demonstrate a variety of methods, including mixed-integer programming, iterative local search, and other optimization techniques. These methods scale poorly with constellation size due to combinatorial complexity and require a priori knowledge of the task list. Some treatments consider aleatoric uncertainty but do not use real-time information to inform future decisions, maintaining the ground-based preplanning approach [8, 9]. Markov decision processes (MDPs) provide a framework for closed-loop decision-making in uncertain environments: Eddy [10] and Harris [11] formalize the Earth-observing spacecraft tasking problem as an MDP, and Herrmann demonstrates the use of deep reinforcement learning (RL) to learn a single-agent autonomous policy [12]. Research into RL in the multiagent setting has not sufficiently explored decentralized planning for constellations with tasks that require cooperation. Some algorithms merge RL with centralized global planning stages [13–15]. Others only consider preventing the duplication of tasks: In [16], the closed-loop nature of a single-agent-trained policy is leveraged in a multiagent setting with communication to deduplicate tasks in a policy that has not inherently learned to coordinate. Li considers RL on a similar tasking problem, in which task deduplication can be achieved through value modification [17].

*Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Dr., Colorado Center for Astrodynamics Research, Boulder, CO, U.S.A. 80303.

†Professor and Department Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Dr., Colorado Center for Astrodynamics Research, Boulder, CO, U.S.A. 80303.

In this work, a constellation tasking problem for coordinated and responsive identification and monitoring of wildfires as they evolve is formulated as an asynchronous decentralized partially observable semi-Markov decision process (Dec-POsMDP), building on prior work considering the single-agent case [18] and the use of single-agent-trained policies in a multiagent setting [16]. Wildfires are selected as the target of interest as there is no prior knowledge about their location or value, necessitating closed-loop planning. Each satellite is equipped with a low-resolution forward-looking scanning instrument that can coarsely identify the locations of wildfires as they appear, and a high-resolution pointing instrument that can image the wildfires in detail to produce information of interest to stakeholders, similar to the architecture in [19]. The goal of the problem is to maximize the value of images collected by the constellation, where image value is a function of the wildfire's intensity at the time of the image, which can only be coarsely determined by the low-resolution instrument: large, rapidly-evolving wildfires are more valuable than small, short-lived ones. A high-fidelity simulation environment is configured using the BSK-RL package for spacecraft tasking RL environments [20, 21].

Given information about the agent's own state and upcoming known wildfires, each agent independently and asynchronously decides onboard which upcoming target to attempt to image next, subject to slew time and visibility constraints inherent to the high-fidelity simulation. Instead of coordinating actions at the policy level, collaboration is achieved through a realistic notion of intersatellite communication defined internally to the MDP, in which information about the previous imaging times and values for each wildfire is shared between satellites at a reasonable interval. In a multiagent training environment, deep RL is used to learn a single closed-loop policy that is executed locally by each agent, implicitly leading to scalability, responsiveness, and robustness. In particular, a modified implementation of proximal policy optimization is used that accounts for variable-duration actions in semi-MDPs [22–24]. A base single-agent policy is trained and then fine-tuned on a variety of constellation architectures, demonstrating that in scenarios with more interaction between agents, the policy can be improved by multiagent training.

2. Problem Formulation

The problem is constructed in three parts: a model of the science objective that the constellation aims to monitor, a description of the satellite capabilities and behaviors, and a formalization of the closed-loop scheduling problem as a MDP. In summary, wildfires appear stochastically and evolve in intensity over time. Satellites can detect these fires with a low-fidelity sensor, but must using a high-fidelity agile-pointing instrument to collect scientifically valuable data. The task is to maximize the value of the data collected by the constellation, in which the value of an image is proportional to the intensity of the fire at the time of the image. Satellite must decide which upcoming fires to image based low-fidelity estimates of value and satellites' previous images of the fires, subject to slew time and data buffer constraints.

2.1. Wildfire Science Objective Model

Wildfires are selected as the objective for the satellite tasking problem due to their unpredictable appearance, rapid evolution, and a motivating scientific and practical value for observing their behavior. A sufficiently detailed model of wildfires is necessary to drive the tasking problem; as such, a generative model for wildfires has been derived from a database of historical wildfire occurrences [25]. With this model, random but realistic distributions of fire instances can be sampled for use in a RL environment.

A multistep approach to generating wildfire instances is taken. First, a global wildfire occurrence rate process is sampled, which gives the rate at which new fires occur (Equation 11). This model is dependent on the time of year being sampled. Based on this rate, fire ignition times $t_{0,i}$ are generated. The location of the fire is then sampled from a global, day-of-year-dependent distribution of fire occurrences (Figure 11). Finally, a duration Δt_i (Equation 14) and total burnt area A_i (Equation 15) are sampled from a joint distribution. Typically, this fire model generates 3000 to 10000 active fires at any given time, usually highly clustered in certain regions of South America and Africa, depending on the time of year.

Each fire is assigned a time-varying intensity function $I_i(t)$, seen in Figure 1. This function is meant to be representative of how "interesting" a fire is at a given time. For many applications, this is reasonably related to how large or active the fire is. As such, this function is arbitrarily defined as

$$I_i(t) = \frac{1}{10}\left|A_i - \frac{4A_i}{\Delta t_i^2}\left(t - t_{0,i} - \frac{\Delta t_i}{2}\right)^2 + G(t)\right|, t \in [t_{0,i}, t_{0,i} + \Delta t_i] \tag{1}$$

where $G(t)$ is a Gaussian process with a mean of 0, a variance $A_i^2/100$, and length scale of $\ell = 50$ seconds. Effectively,

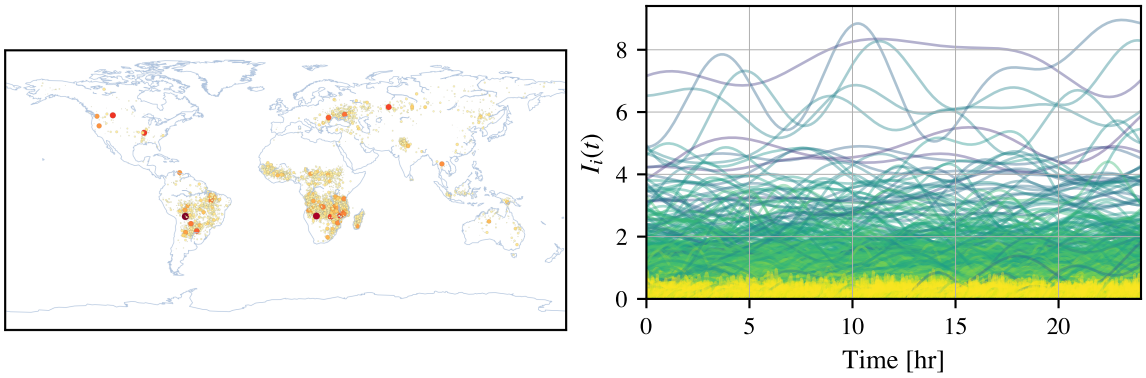

Fig. 1 The model is instantiated for a one-day duration. (Left) Map of fire locations; marker size and color indicates fire intensity. (Right) Change in fire intensity over time; color indicates fire duration.

this is a parabola with roots at the start and end times of the fire and a peak proportional to the area, plus a random walk. For a specific application, this function would be derived from expert knowledge about which qualities of a fire are most important to observe for the given use case.

Detailed model parameters are given in the appendix.

2.2. Satellite Architecture

An agile Earth-observing satellite (AEOS) is modeled with two instruments: the agile pointing instrument used to collect scientifically valuable data, and an always-on lookahead scanning instrument that detects upcoming fires along with a rough estimate of their intensity. This operations scheme is similar to that in [26] as well as other dynamic tasking architectures. This architecture is key for demonstrating the in-the-loop planning capabilities of RL-based policies, as the agent does not have any prior knowledge of the task locations or values, as opposed to in previous work in which the satellite is tasking over a fixed-location, fixed-value request list [18].

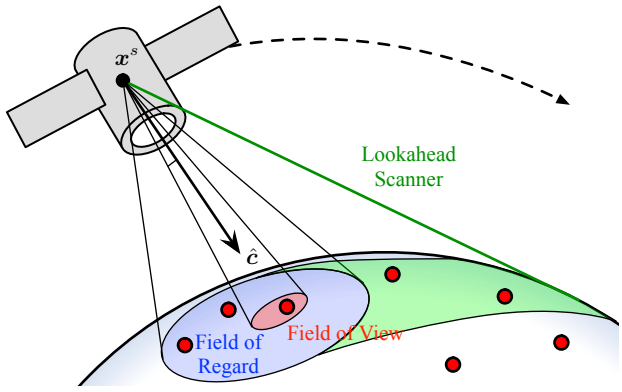

Fig. 2 The satellite's field of view when collecting an image, field of regard for possible collects, and lookahead for upcoming fires.

The lookahead scanning instrument identifies fires from the satellite's current location up to the horizon, or approximately a 7 minute horizon of upcoming along-track opportunities. This instrument provides knowledge of fire i's location and a coarse estimate of the fire intensity:

$$\tilde{I}_i(t_{\text{scan}}) = \lceil I_i(t_{\text{scan}}) \rceil \tag{2}$$

The purpose of this instrument is to provide the agent with information about possible upcoming tasks.

The agile pointing instrument is used to collect scientifically valuable data, which is the objective of the mission. When the instrument, which has a body-fixed boresight direction of $\hat{\boldsymbol{c}}$ is pointing at a fire within some pointing threshold

$\delta\theta_{\max}$ (i.e. the field of view in Figure 2) and rate threshold $\delta\omega_{\max}$, an image is collected, yielding an accurate value for the fire's current intensity $I_i(t_{\text{image}})$ and a proportional reward towards the maximization objective. The agile instrument is pointed using the satellite's attitude dynamic control system, which implement an asymptotically stable pointing controller driven by reaction wheels [27]. As a result, transition times between imaging different targets are nonzero and depend on the slew time required to point the instrument. Fires can only be imaged if they are within an elevation limit ϕ of the satellite's current position corresponding to a 500 km radius field of regard.

Furthermore, each satellite has a finite amount of data storage in the buffer $B_{\max}$. Only 50 images may be stored at a time. When within a 45° elevation of one of seven ground stations (Boulder, Merritt Island, Singapore, Weilheim, Santiago, Dongara, and Hawaii), satellites can downlink data at a rate of one image per second, freeing buffer space for new image collection. If a satellite attempts to collect an image while the storage is full, it is unsuccessful.

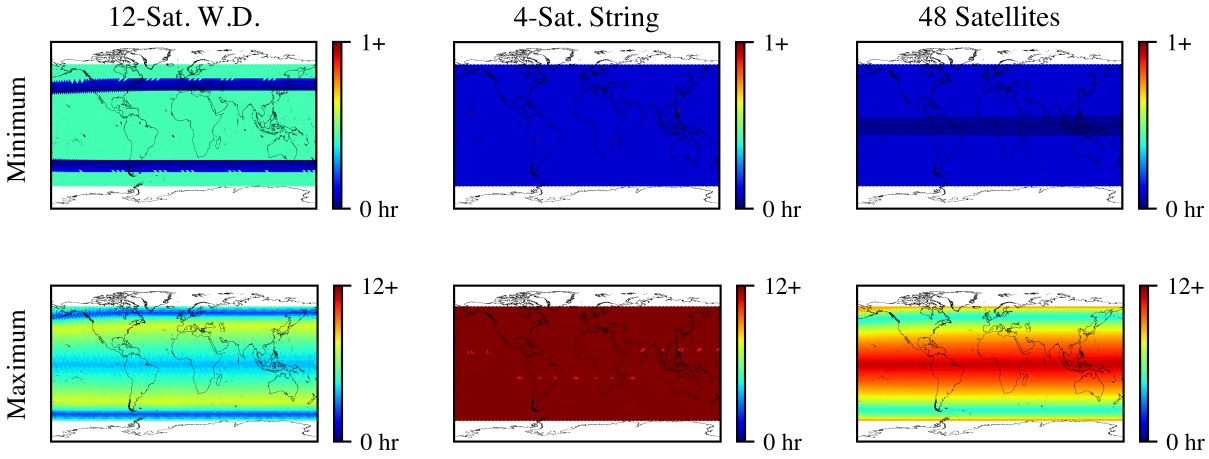

Fig. 3 Minimum (bottom) and maximum (top) revisit times [hr] for the (left) 12-satellite Walker-delta, (middle) string constellation, and (right) 48-satellite constellations.

Satellites are placed in 800 km, 60° inclination circular orbits; this inclination allows the satellites to cover the majority of the latitudes over which wildfires occur. Three multisatellite constellations are considered in this work: A 3-plane × 4-satellite-per-plane Walker-delta constellation, a 4-satellite "string" constellation with 20° true anomaly spacing, and a 48-satellite constellation made of two planes of 24 satellites each. The respective maps of coverage and revisit frequency for each constellation is given in Figure 3, generated using the Tool for Early Mission Planning and Observation (TEMPO)* [28, 29].

2.3. Information and Objective

In order to define an objective for the tasking problem and inform the agents' decision-making, each satellite records two logs of information based on the observations made by the instruments:

- **Latest Scan Log:** For each fire i that has been scanned with the low-fidelity scanning instrument, a tuple containing the fire index, the time of scanning, and the low-fidelity estimate of intensity $[i, t_{\text{scan}}, \tilde{I}_i(t_{\text{scan}})]$ is recorded for the latest scan. This tuple is overwritten whenever new scanning information is collected about the fire.
- **All Collect Log:** Whenever a fire is collected by the agile imaging instrument (i.e. the pointing thresholds are met and storage is available in the data buffer $B < B_{\max}$), a tuple containing the fire index, the time of the collect, and the high-fidelity measurement of the intensity $c = [i, t_{\text{col}}, I_i(t_{\text{col}})]$ is appended to a list of all collects C.

Note that the size of these logs are unrelated to the data buffer fullness B; they are instead treated as metadata with a minimal storage requirement.

*https://github.com/jsipps26/TEMPO

The objective of the tasking problem is to maximize the value of the images collected by the constellation:

$$\text{maximize} \sum_{c \in C} r(c) \tag{3}$$

Per-collect reward r is assumed to be proportional to the intensity of the fire at the time of the collect for this application; the intensity function and/or reward function r could be arbitrarily redefined if different behavior was desired. To prevent overimaging of a single request, a linear penalty on collect frequency is applied. The reward function is

$$r(c) = \begin{cases} \frac{t-t'}{\tau} I_i(t_{\text{col}}) & \text{if } \exists\ c' \in C \text{ s.t. } t - t' < \tau \\ I_i(t_{\text{col}}) & \text{else} \end{cases} \tag{4}$$

where $\tau = 30$ minutes is the penalty duration.

In multiagent settings, each satellite maintains its own logs of scans and collects. To enable coordination between satellites, a communication mechanism is defined that allows satellites to share information about their logs. Intersatellite communication is defined as the exchange of log data: For the scanning log, newly identified fires from other satellites are added, and existing fires are updated with the latest intensity estimate. For the collect log, new collects completed by other satellites are added. It is assumed that intersatellite communication is free and instantaneous. Communication is performed every five minutes within the simulation. This environment-implicit communication avoids the need for more complex multiagent reinforcement learning (MARL) techniques that require policies to be able to communicate prior to deciding on any action; it also allows for communication to be defined in an arbitrary satellite-realistic way.

2.4. Problem Formalization

To approach the problem with RL, the wildfire monitoring task is formalized as a Dec-POsMDP. A MDP is a generic framework for expressing tasking problems, in which an agent selects actions $a \in \mathcal{A}$ based on the current state $s \in \mathcal{S}$ of an environment; in turn, the environment returns a new state s' and a scalar reward r [30]. The objective of an agent is to maximize the sum of future rewards.

Partially-observable Markov decision processs (POMDPs) extend the MDP framework to include partial observability, in which the agent does not have full knowledge of the environment state. This may be due to epistemic limitations or an excess of functionally irrelevant states in the simulator; both are true in the environment. Decentralized POMDPs (dec-POMDPs) extend this notion to a multiagent setting, in which each agent has its own partial observation of the environment and must select its own action based on its own observation [31].

In a semi-Markov decision process (sMDP), each step has a duration associated with it. For example, an action that yields reward after a longer duration step will see that reward more heavily discounted than an action that yields reward quickly. When applied to the multiagent case, this requires agents to make decisions asynchronously, as the duration of each agent's step may be different.

The MDP is implemented using BSK-RL, a tool for generating high-fidelity, modular spacecraft tasking RL environments[†] [21]. Underlying models include rigid body dynamics with reaction-wheel based control, execution of flight software for each flight mode, and realistic orbital and planetary dynamics. Frameworks to represent data, communication, and the objective are also included in the package. To formalize the wildfire monitoring task as a Dec-POsMDP, the following elements are defined, reflecting the generative model provided by the simulation environment:

- **State Space $\mathcal{S}$:** The state space is the complete space of simulator states required to maintain the Markov assumption. This includes directly observable variables such as each spacecraft's dynamic state and known information about fires, unknown information about fires, and variables required for simulation such as controller integrator states. Practically, only a subset of the state is relevant to the scheduling problem, which is exposed through the observation function.
- **Observation Space O and Observation Probability Function Z:** The observation space O for each agent consists of a deterministic selection of dimensions from the state space and transformations thereof. The selected elements are those presumed to be relevant to decision-making for the problem, based on expert knowledge and experimentation. Informed by prior work [18], the observations for each satellite in this environment include information about the satellite's dynamic state and the next $N = 32$ upcoming known fires, as given in Table 1. In

[†] `https://avslab.github.io/bsk_rl/`

Quantity	Norm.	Dim.	Description
${}^{\mathcal{H}}\boldsymbol{\omega}_{\mathcal{BH}}$	0.03 rad/s	3	Hill-frame body angular rate
${}^{\mathcal{H}}\hat{\boldsymbol{c}}$	-	3	Hill-frame instrument pointing direction
${}^{\mathcal{E}}\boldsymbol{r}_{\mathcal{BE}}$	r_E	3	Earth-fixed position, Earth radius-normalized
${}^{\mathcal{E}}\boldsymbol{v}_{\mathcal{BE}}$	v_{orb}	3	Earth-fixed velocity, orbital velocity-normalized
B	$B_{\max}$	1	Data buffer fill level
$[t^o_{\text{down}}, t^c_{\text{down}}]$	T_{orb}	2	Next downlink opportunity open and close time
${}^{\mathcal{H}}\boldsymbol{r}_i$	800 km	$3 \times N$	Hill-frame position of next N known fires
$\delta\theta_i$	$\pi/2$ rad	$1 \times N$	Pointing error of next N known fires
$[t^o_i - t, t^c_i - t]$	T_{orb}	$2 \times N$	Opportunity open and close of next N known fires
$[t_{\text{scan}}, \tilde{I}_i(t_{\text{scan}})]$	$[T_{\text{orb}}, 1]$	$2 \times N$	Latest scan information of next N known fires
$[t_{\text{col}}, I_i(t_{\text{col}})]$	$[T_{\text{orb}}, 1]$	$2 \times N$	Latest collect information of next N known fires

Table 1 Elements in each satellite's observation o and their normalization constants.

particular, the information about fires draws from the latest scan log and all collect log, which contain information collected by the satellite and communicated from other satellites. Observation elements are normalized to fall approximately in $[-1, 1]$, which improves the performance of deep reinforcement learning (DRL) algorithms.

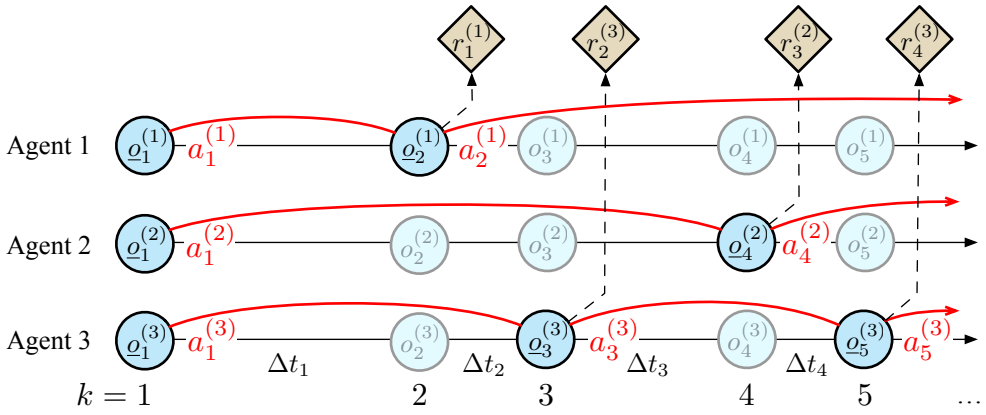

Fig. 4 The asynchronous decentralized semi-MDP framework.

- **Action Space $\mathcal{A}$:** Each satellite has an action space corresponding to different flight software modes and settings. The satellite has $N = 32$ imaging actions $a_{\text{im},n}$ that task the satellite with collecting an image of the n^{th} upcoming fire. The availability of an action does not imply feasibility: a fire may leave the field of regard before the satellite can slew to and settle the instrument in the pointing direction. This action is executed until either collection is successful, or the fire leave the field of regard (i.e. collection is unsuccessful). Additionally, the satellite has a downlink action a_{down} which frees space in the data buffer for new collects if the satellite is within range of a ground station; this action is attempted for one minute.

  In multiagent settings, agents may act asynchronously (Figure 4) [23]. This implies two equivalent interpretations of the action space: 1) the action space at a given step is the product of action spaces only of agents that have completed their current task; or 2) the action space of all agents that are mid-task when another agent requires a new action is a single "continue current action" action.

- **Transition Probability Function $T(s'|s, a)$ and Per-Agent Step Duration Function $F(\Delta t|s, a, s')$:** Transitions are deterministic and generated by the simulator. The simulator propagates until at least one agent requires a new action (i.e. one of the conditions for an action to complete has be satisfied). In the sMDP framework, the transition time Δt, the time elapsed between the previous state and the current state (Figure 4), is used in learning.

- **Reward Function $R(s, a, s')$:** The reward function is defined as the sum of the rewards of all collects completed during the step. The reward for each collect is proportional to the intensity of the fire at the time of the collect, minus a penalty for high-frequency reimaging, as described above. Mathematically,

$$R(s, a, s') = \sum_{c \in C' \setminus C} r(c) \tag{5}$$

  where C' is the set of all collects completed after the step and C is the set of all collects completed before the step; thus, the reward is the sum of the rewards of all collects completed during the step.

With the MDP defined, RL can be applied to the problem to learn a policy that maximizes the value of the data collected by the constellation.

3. Methodology

A brief overview of RL is given, followed by the necessary modifications to RL algorithms to account for asynchronous, variable-duration actions in the Dec-POsMDP framework.

3.1. Reinforcement Learning

RL is a strategy for solving a MDP by learning a policy (that is, a mapping of states to actions $\pi(s) = a$) that maximizes the expected discounted sum of future rewards. Discounting multiplies future rewards k steps in the future by a factor $\gamma^k, \gamma \in [0, 1)$ to ensure that the sum converges for infinite horizons. This discounted sum is called value:

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma r_k | s_{k+1} \sim T(s_k, \pi(s_k))\right] \tag{6}$$

where $r_1, ...r_k$ is the sequence of rewards received by following policy π from state s. The policy optimization problem is thus

$$\pi^* = \arg\max_\pi V^\pi(s) | s \in S \tag{7}$$

Reinforcement learning finds a good policy by iterative improvement of the policy while interacting with the environment. The algorithm does not have knowledge of the MDP's properties. Rather, it must learn by interacting with the environment to find what behaviors lead to beneficial results, trading off exploration of new behaviors with exploitation of known good behaviors.

In this work, proximal policy optimization (PPO) is selected as the algorithm used to generate policies, given its strong performance over a wide domain of problems [22]. PPO is a DRL algorithm that represents the policy with a neural network and updates the policy with clipped policy gradient updates. Prior work has demonstrated that this algorithm performs well across a variety of spacecraft tasking problems [12, 18].

3.2. Asynchronous Semi-Markov Decision Processes Discounting

In the sMDP framework, actions have a duration associated with them. This duration can be used to properly account for the time-cost of actions. In this paper, the convention that reward is yielded at the end of a step is assumed (as opposed to having some distribution throughout the step). Since PPO calculates the value of rollouts encountered in training, properly computing value for the sMDP accounting for variable-duration timesteps is beneficial.

3.2.1. Single-Agent

In the single-agent case, the agent takes an action at every step. The value function is thus

$$V(s) = \gamma^{\Delta t_1} r_1 + \gamma^{\Delta t_1 + \Delta t_2} r_2 + ... \tag{8}$$

$$= \sum_{k=1}^{\infty} \gamma^{\sum_{i=1}^{k} \Delta t_i} r_k \tag{9}$$

This equation can be incorporated into PPO's generalized advantage estimation (GAE) calculation to be used in learning, by substituting instances of value in the GAE equation. However, the GAE λ parameter continues to average over steps instead of time.

3.2.2. Multi-Agent

In the multi-agent case, we assume that agents' actions are not highly coupled; agents do not need to anticipate when other agents will complete their actions, and completion of one agent's action should not interrupt another agent's action, as shown in Figure 4. Interaction and communication between agents is implicit to the environment. In effect, each agent can be treated as a single-agent sMDP, training only on observations $\underline{o}$ that occur when the agent requires retasking.

To accomplish this, each agents' rollouts are condensed by concatenating non-acting steps. For example, the rollout used in training for agent 3 in Figure 4 would become

$$(\underline{o}_1^{(3)}, a_1^{(3)}, \Delta t_1 + \Delta t_2, r_2^{(3)}), (\underline{o}_3^{(3)}, a_3^{(3)}, \Delta t_3 + \Delta t_4, r_4^{(3)}), (\underline{o}_5^{(3)}, a_5^{(3)}, \Delta t_5 + ... + \Delta t_k, r_k^{(3)}), ... \tag{10}$$

by using the observation and action at the time of retasking, a step duration consisting of the sum of the multiagent step durations until the agent retasks, and the reward at the time of retasking.

3.3. Multiagent Training

In both single and multiagent cases, as single per-agent policy is trained for all satellites, with all agents contributing to the training of a single policy. For constellations, this avoids a key challenge in multiagent RL for decentralized agents: the nonstationarity of the super-environment consisting of the base environment and all other agents as their individual policies change. All experience is used for training a single policy, and as such the possibility for feedback loops between different policies is eliminated. This approach is also beneficial for the scalability of the method to constellations of different sizes, as the same policy can be deployed on any agent. Because of the inclusion of communication implicitly in the environment, the policy is not explicitly dependent on a certain agent count.

There is no inherent limitation preventing the training of different policies for different agents (i.e. differentiating between roles for satellites) using this asynchronous sMDP framework; however, that may introduce the aforementioned issues with nonstationarity in training, so it is not explored in this work.

Because the multiagent environment has additional computational overheads, a fine-tuning approach is taken for training. A policy is trained in the single agent environment until training converges. Then, the policy is fine-tuned with a second round of training performed in the multiagent scenario. All agents' experience are used to training the single shared policy, which is initialized the the single-agent base policy. This approach is reasonable because the bulk of the agent's strategy can be reasonably expected to be the same between the single and multiagent cases; the fine-tuning allows for the agent to learn any improvements to the policy the leverage the interaction between agents.

The policy is trained using the RLlib implementation of PPO, with custom connectors configured for asynchronous sMDP training [24]. Key hyperparameters are $\gamma = 0.9997$ (with Δt in units of seconds), learning rate 3×10^{-5}, and a training batch size of 6000 steps/agent, as determined by a hyperparameter search. Training is performed on 30 parallel copies of the environment, so the learner receives 200-step rollouts, on average.

4. Results

The single agent environment is trained for 4 days of wall-clock time, or a total of 6.4M steps across the 30 parallel workers; this corresponds to 17k one-day simulation time episodes. Then, it is fine-tuned on the 4-agent "string" constellation for an additional wall-clock day (1.3M steps) and separately on the 12-agent Walker-delta constellation for an additional wall-clock day (0.7M steps). Each training time was sufficient to reach a point at which the policy was no longer improving. First, the single agent policy is benchmarked in a single agent scenario. Then, each policy is compared when deployed on the two constellations used for fine-tuning. Finally, the scalability of the policy is tested by deploying it on a 48-satellite constellation.

4.1. Single Agent Performance

First, the base policy is evaluated in a single-agent environment. Figure 5 compares the available rewards in the environment to the reward collected by the policy across 150 day-long trials. It is evident that the policy is primarily relying on the low-resolution estimates of value to select fires to collect, as seen by the step-function-like ratio of collected to available rewards. This is expected, as the single satellite rarely re-encounters the same fire, and thus has little opportunity to learn from fire intensities obtained from previous collections.

Figure 6 looks at a single instance of the policy in action. Reflecting the aggregate performance (Figure 5), the policy is seen to tend toward collecting high-value fires. The policy is also aggressive in regard to filling the data buffer, opting to almost always have a full buffer before the next downlink opportunity. This behavior makes sense in relation

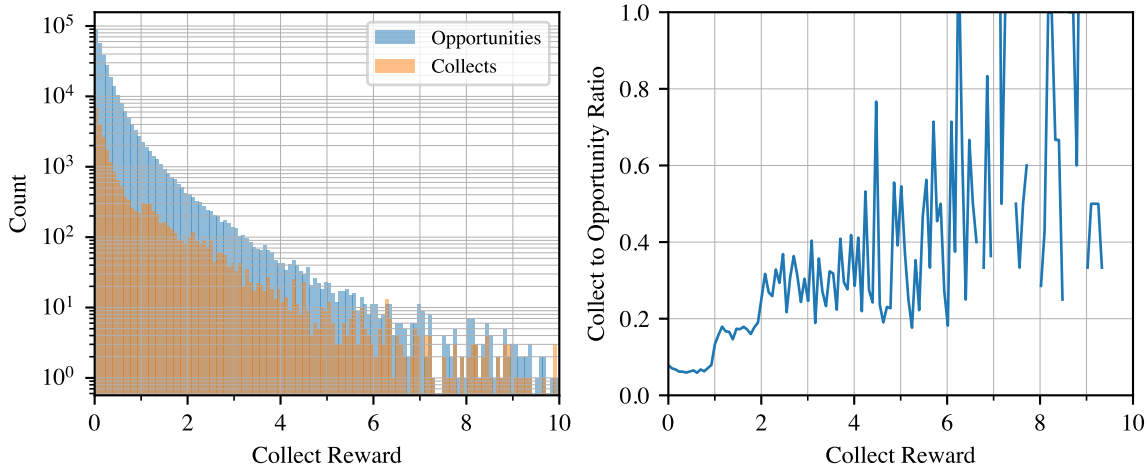

Fig. 5 In the single satellite environment, distribution of rewards obtained by the policy versus those available in the environment.

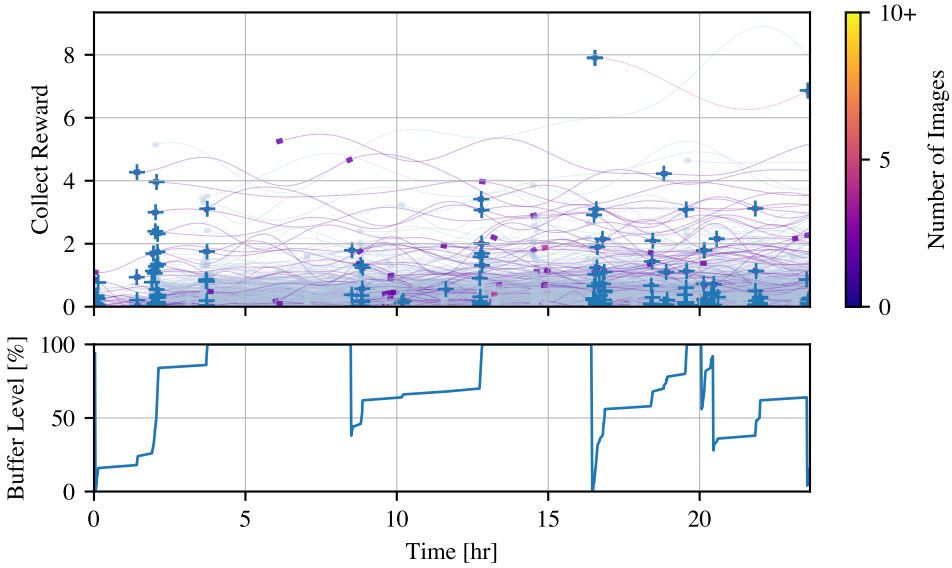

Fig. 6 An instance of the base policy in the single-satellite environment. (Top) Each line corresponds to a fire intensity, with wide spots indicating a collection opportunity; markers are successful image collections; (bottom) satellite data buffer fullness over time.

to the reward model: the downside of not collecting images as much as possible is greater than the upside of leaving the buffer with open space for a collect of a rarer high-value fire.

4.2. Multiagent Fine-Tuning Performance

The two fine-tuned policies are evaluated against the base policy when deployed on the string and Walker-delta constellations in Figure 7. The string fine-tuned policy shows markedly improved performance (on average, an 18% improvement over the base policy) when deployed on the string constellation. This can be attributed to the relatively high interaction between close satellites, which allows the policy to leverage the information collected by other satellites to make better decisions. However, the Walker-delta fine-tuned policy does not improve over the base policy. It may be that improvement over the base policy on the Walker-delta constellation is not easily attainable, as the large spacing results in an environment that is effectively a collection of single-agent environments without enough interaction between agents

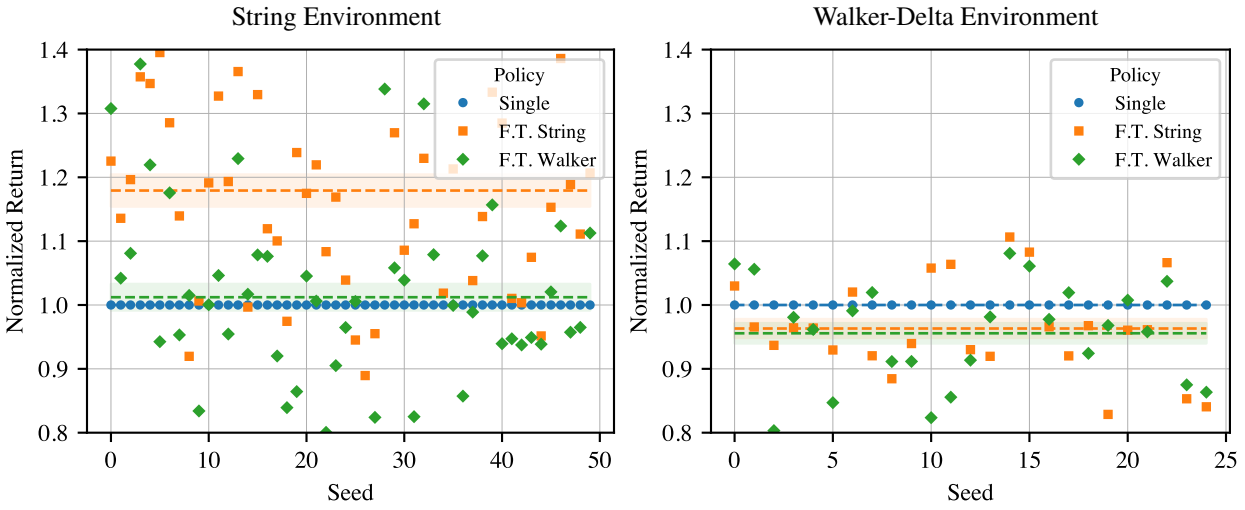

Fig. 7 Deployment of each policy on (left) string constellation and (right) 12-satellite Walker-delta constellation. Performance is normalized by the single-agent policy's return for that case.

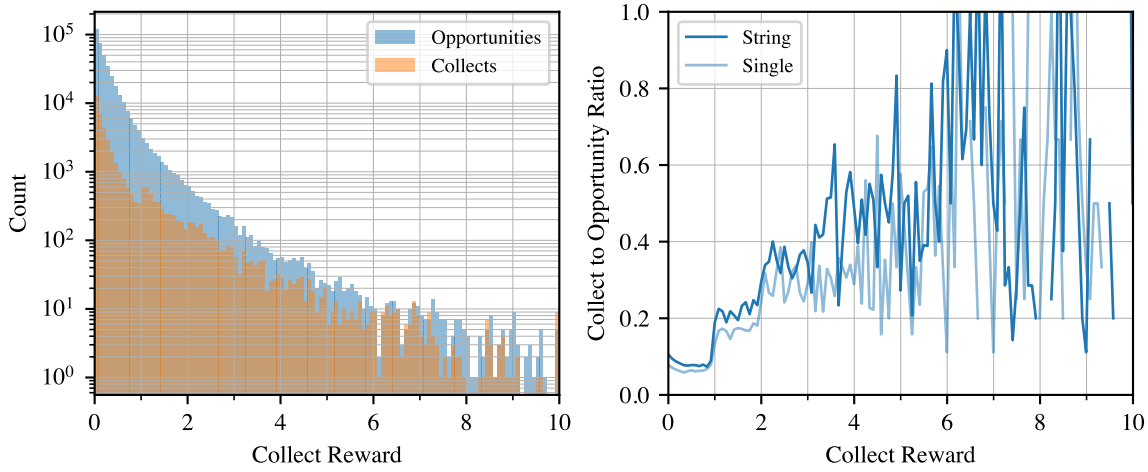

Fig. 8 In the string constellation environment, distribution of rewards obtained by the string fine-tuned policy versus those available in the environment. Right plot is compared to the single agent policy results from Figure 5.

to be exploitable.

The key to the improved performance of the string fine-tuned policy becomes evident in Figure 8. While the single agent policy experienced relatively few cases of reencountering the same fire multiple times and thus relied on the low fidelity estimate of intensity to select targets, the multiagent fine-tuned policy gained experience in a domain where previously encountered fires could be reimaged. As a result, the fine-tuned policy learned to reimage fires based on a high observed intensity in previous images, as is evident for fires of value greater than one in the right plot of Figure 8: the string fine-tuned policy collects a higher proportion of high-value targets than the single agent policy.

The behavior of the string fine-tuned policy is shown in Figure 9. Many of the behaviors are qualitatively similar to the single-agent base policy, such as the tendency to fill the data buffer before downlink opportunities and a bias towards high-value fires. This constellation architecture does allow for more rapid recollection of the same fire, as satellites have a roughly 5-minute difference in revisit times, which is lower than the 30-minute rapid recollection penalty in Equation 4. Occasionally, the agents opt to collect a fire multiple times in quick succession, since the behavior is not strongly penalized: a high-value fire with an 80% revisit penalty is equally attractive as a low-value fire. If rapid reimaging behavior was strongly undesired, the penalty could be increased to discourage it. A positive aspect of the reimaging behavior is evident from the upper subplot: higher interest fires tend to be imaged multiple times over the

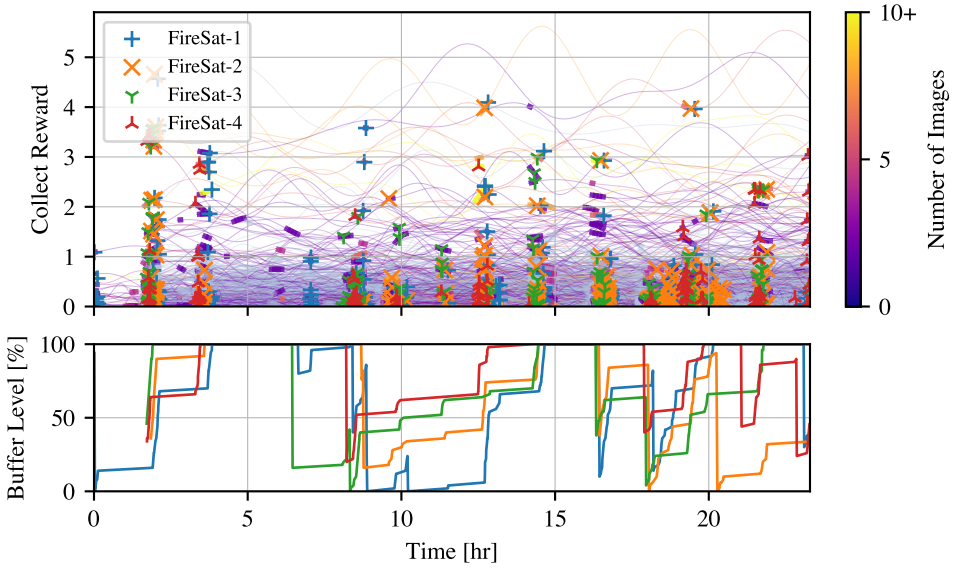

Fig. 9 An instance of the string fine-tuned policy in the string constellation environment. (Top) Each line corresponds to a fire intensity, with wide spots indicating a collection opportunity; markers are successful image collections; (bottom) satellite data buffer fullness over time.

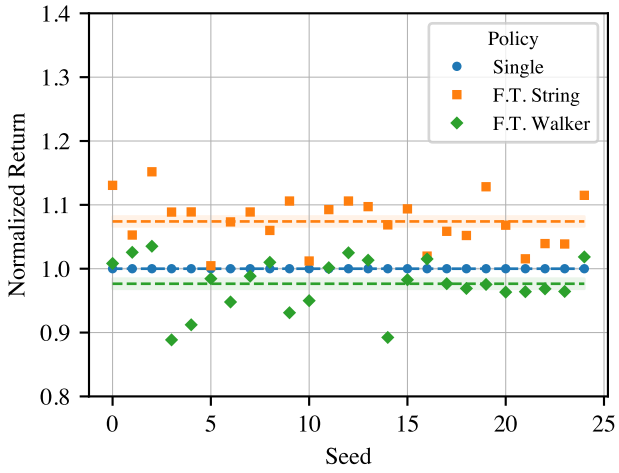

Fig. 10 Relative performance of each policy when deployed on the 48-satellite constellation.

course of a day, which aligns with the objective of the tasking problem. This indicates that the selected reward function effectively represents the abstract goal of the mission, even with the constellation being clearly oversubscribed with the number of satellites relative to the number of fires.

4.3. Constellation Scalability

Finally, to test the scalability of the method, each policy is evaluated on the 48-satellite constellation. Directly fine-tuning on this constellation would be prohibitively expensive in the high-fidelity simulation environment. When comparing on 6-hour long episodes in Figure 10, the string fine-tuned policy again outperforms the base policy, by about 8%. Since the relative satellite spacing in the 4-satellite string constellation and the 48-satellite constellation are similar, this result makes sense: the behavior learned to optimize between satellites in the string constellation is still effective in the larger constellation. The Walker-delta fine-tuned policy, however, does not improve over the base policy, as in the 12-satellite constellation. This result suggests that the Walker-delta constellation is not well-suited to

the reimaging behavior learned by the fine-tuned policy, as the large spacing between satellites makes it difficult to leverage the information collected by other satellites.

5. Conclusions

The strengths of RL for spacecraft tasking are highlighted on the wildfire application, specifically formulated for the asynchronous Dec-POsMDP framework introduced by this paper. Closed-loop planning of opportunistically available targets is performed with low computational cost, as opposed to the open-loop methods typically used in spacecraft scheduling. The per-agent policy also allows for scalability to constellations of any size. If properties of the deployment constellation are known, the policy can be fine-tuned on a similar constellation to further improve performance by leveraging constellation-specific interactions.

The results show raise questions to be addressed in future work: In particular, while this task can be better completed with some multiagent interaction, it is not essential for success. Reformulating the objective in a way that necessitates collaboration between agents, such as with a different reward function and different observability of states, could lead to a more challenging problem that would require multiagent learning to solve. Theoretical analysis of the asynchronous Dec-POsMDP framework could also be beneficial to understand the conditions under which multiagent learning possible or limited.

Acknowledgments

This work is supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) grant, 80NSSC23K1182. This work utilized the Alpine high-performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (award 2201538).

Appendix A: Generative Wildfire Model

A random process model for the global wildfire occurrence intensity function $\lambda(t, \omega)$ in new fires per day is given by the sum of a periodic function $\lambda_f(t)$ to capture seasonal effects and a Gaussian process with exponential covariance $\lambda_e(t, \omega)$ to capture the stochastic nature of the process:

$$\lambda(t, \omega) = \lambda_\mu + \lambda_f(t) + \lambda_e(t, \omega) \tag{11}$$

The Fourier series in the typical form

$$\lambda_f(t) = \sum_i a_i \cos(2\pi f_i t + \phi_i) \tag{12}$$

is fit to the daily new fire occurrence data from 2002 to 2018 using a fast Fourier transform (FFT). The Fourier model parameters are given in Table 2.

Table 2 Fourier series fit parameters.

f [year^{-1}]	a [fires/day]	ϕ [rad]
λ_μ	2972.9	-
1.0	796.2	2.109
2.0	570.2	-1.188
3.0	318.7	0.618

A Gaussian process $\lambda_e(t, \omega)$ with exponential covariance is used to model the stochastic nature of the global occurrence intensity; specifically, this term should capture the statistics of the residuals of the data relative to the Fourier term $\lambda_f(t)$. The exponential covariance function is given by

$$C_{XX}(\Delta t) = \sigma^2 \exp\left(-\frac{\Delta t}{\ell}\right). \tag{13}$$

with $\sigma = 945.8$ fires/day, $\ell = 2.935$ years. Poisson point process sampling is used to generate ignition times for fires given a realization of the intensity function at the time of year being generated.

Fire locations are generated from a time-dependent distribution of fire occurrences $f_X(\boldsymbol{x}|t)$, which is derived from the data set. Samples of the distribution are shown in Figure 11.

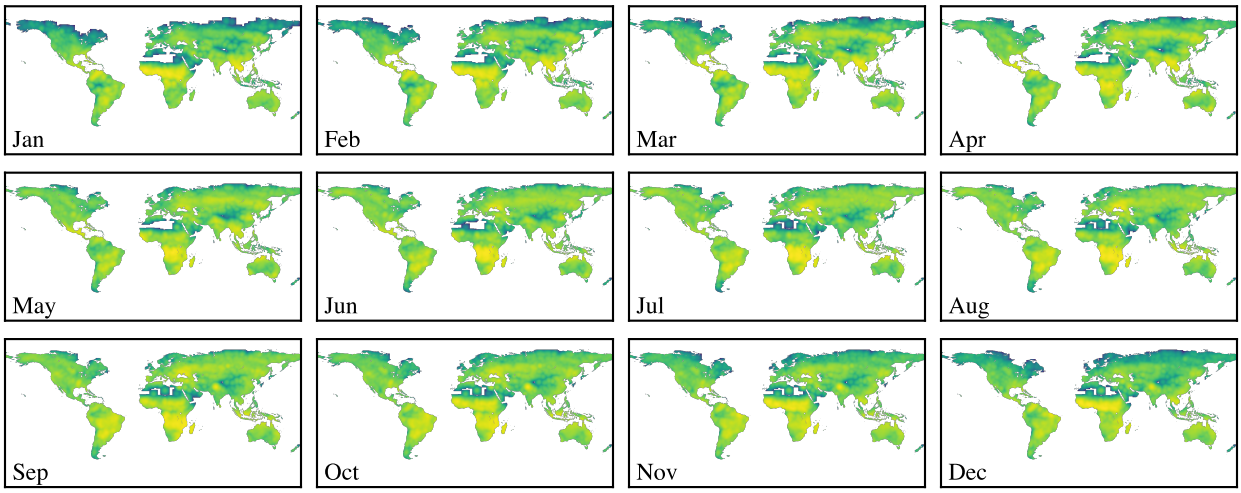

Fig. 11 Probability density function $f_X(\boldsymbol{x}|t)$ of fire occurrences by season; blue: fewer fires, yellow: more fires.

Marginal distributions for fire duration and area are fit to the entire fire dataset, as no strong location or time dependence is observed. The duration Δt_i is sampled from a Pareto distribution

$$\Delta t_i \sim 157.5\mathrm{Pr}(a = 133.1) \text{ days} \tag{14}$$

while the area A_i is sampled from a normal distribution dependent on the duration

$$A_i \sim \mathcal{N}(\mu = 0.193\Delta t_i^{1.784} + 0.755, \sigma = 0.234\Delta t_i^{2.234} + 0.989)\mathrm{km}^2 \tag{15}$$

All values are fit to a database containing global fire instances between 2002 and 2018, that includes information on ignition date, burning area, and duration [25].

References

[1] G. Picard, C. Caron, J.-L. Farges, J. Guerra, C. Pralet, and S. Roussel, "Autonomous Agents and Multiagent Systems Challenges in Earth Observation Satellite Constellations," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2021, pp. 39–44.

[2] M. Seablom, J. L. Moigne, S. Kumar, B. Forman, and P. Grogan, "Real-Time Applications of the Nasa Earth Science "New Observing Strategy"," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. Kuala Lumpur, Malaysia: IEEE, Jul. 2022, pp. 5317–5320.

[3] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3881–3892, Sep. 2021.

[4] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, vol. 6, no. 5, pp. 367–381, Sep. 2002.

[5] S. Nag, A. S. Li, and J. H. Merrick, "Scheduling algorithms for rapid imaging using agile Cubesat constellations," *Advances in Space Research*, vol. 61, no. 3, pp. 891–913, Feb. 2018.

[6] D.-H. Cho, J.-H. Kim, H.-L. Choi, and J. Ahn, "Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation," *Journal of Aerospace Information Systems*, vol. 15, no. 11, pp. 611–626, Nov. 2018.

[7] J. Kim, J. Ahn, H.-L. Choi, and D.-H. Cho, "Task Scheduling of Agile Satellites with Transition Time and Stereoscopic Imaging Constraints," *Journal of Aerospace Information Systems*, vol. 17, no. 6, pp. 285–293, Jun. 2020.

[8] C. G. Valicka, D. Garcia, A. Staid, J.-P. Watson, G. Hackebeil, S. Rathinam, and L. Ntaimo, "Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty," *European Journal of Operational Research*, vol. 275, no. 2, pp. 431–445, Jun. 2019.

[9] X. Wang, Y. Gu, G. Wu, and J. R. Woodward, "Robust scheduling for multiple agile Earth observation satellites under cloud coverage uncertainty," *Computers & Industrial Engineering*, vol. 156, p. 107292, Jun. 2021.

[10] D. Eddy and M. Kochenderfer, "Markov Decision Processes For Multi-Objective Satellite Task Planning," in *2020 IEEE Aerospace Conference*. Big Sky, MT, USA: IEEE, Mar. 2020, pp. 1–12.

[11] A. Harris, T. Teil, and H. Schaub, "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," in *AAS Spaceflight Mechanics Meeting*, Maui, Hawaii, 2019-01-13/2019-01-17.

[12] A. Herrmann and H. Schaub, "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 5, pp. 1–13, Oct. 2023.

[13] R. Tharmarasa, A. Chatterjee, and Y. Wang, "Closed-Loop Multi-Satellite Scheduling Based on Hierarchical MDP," in *22nd International Conference on Information Fusion*, Ottawa, 2019-07-02/2019-07-05.

[14] Y. He, L. Xing, Y. Chen, W. Pedrycz, L. Wang, and G. Wu, "A Generic Markov Decision Process Model and Reinforcement Learning Method for Scheduling Agile Earth Observation Satellites," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 3, pp. 1463–1474, Mar. 2022.

[15] A. K. Saeed, F. Holguin, A. S. Yasin, B. A. Johnson, and B. M. Rodriguez, "Multi-Agent and Multi-Target Reinforcement Learning for Satellite Sensor Tasking," 2024.

[16] M. Stephenson, L. Mantovani, and H. Schaub, "Intent Sharing for Emergent Collaboration in Autonomous Earth Observing Constellations," in *AAS GN&C Conference*, Breckenridge, CO, Februrary 2, 2024.

[17] P. Li, H. Wang, Y. Zhang, and R. Pan, "Mission planning for distributed multiple agile Earth observing satellites by attention-based deep reinforcement learning method," *Advances in Space Research*, p. S0273117724005465, Jun. 2024.

[18] M. Stephenson and H. Schaub, "Reinforcement Learning For Earth-Observing Satellite Autonomy With Event-Based Task Intervals," in *AAS Rocky Mountain GN&C Conference*, Breckenridge, CO, Feb. 2024.

[19] R. S. Schaefer and P. T. Grogan, "Collaborative Constellation Analysis Framework for Wildfire Observing Missions," *IEEE*, 2024.

[20] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, vol. 17, no. 9, pp. 496–507, Sep. 2020.

[21] M. A. Stephenson and H. Schaub, "BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking," in *75th International Astronautical Congress*. Milan, Italy: IAF, Oct. 2024.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017.

[23] K. Menda, Y.-C. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. Wolpert, "Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1259–1268, Apr. 2019.

[24] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," Jun. 2018.

[25] T. Artés, D. Oom, D. De Rigo, T. H. Durrant, P. Maianti, G. Libertà, and J. San-Miguel-Ayanz, "A global wildfire dataset for the analysis of fire regimes and fire behaviour," *Scientific Data*, vol. 6, no. 1, p. 296, Nov. 2019.

[26] A. Candela, J. Swope, and S. A. Chien, "Dynamic Targeting to Improve Earth Science Missions," *Journal of Aerospace Information Systems*, vol. 20, no. 11, pp. 679–689, Nov. 2023.

[27] H. Schaub and S. Piggott, "Speed-constrained three-axes attitude control using kinematic steering," *Acta Astronautica*, vol. 147, pp. 1–8, Jun. 2018.

[28] J. Sipps and L. Magruder, "Envelope-Based Grid-Point Approach: Efficient Runtime Complexity for Remote Sensing Coverage Analysis," *Journal of Spacecraft and Rockets*, vol. 60, no. 4, pp. 1072–1084, Jul. 2023.

[29] ——, "Virtual Focal Plane Dynamics for Generalized Remote Sensing Coverage Analysis," in *2024 IEEE Aerospace Conference*, Mar. 2024, pp. 1–14.

[30] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, second edition ed., ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts London, England: The MIT Press, 2018.

[31] M. J. Kochenderfer, *Algorithms for Decision Making*. Massachusetts Institute of Technology, 2022.