Schedule Earth Observation satellites with Deep Reinforcement Learning

Adrien Hadj-Salah, Rémi Verdier, Clément Caron, Mathieu Picard, Mikaël Capelle

Schedule Earth Observation satellites with Deep Reinforcement Learning

Adrien Hadj-Salah[1,2], Rémi Verdier[1], Clément Caron[1,2], Mathieu Picard[1,2], Mikaël Capelle[1]
[1]IRT Saint-Exupéry [2]Airbus Defence & Space
{adrien.hadj-salah, remi.verdier, clement.caron, mathieu.picard, mikael.capelle}@irt-saintexupery.com

Abstract

Optical Earth observation satellites acquire images world-wide, covering up to several million square kilometers every day. The complexity of scheduling acquisitions for such systems increases exponentially when considering the interoperability of several satellite constellations together with the uncertainties from weather forecasts. In order to deliver valid images to customers as fast as possible, it is crucial to acquire cloud-free images. Depending on weather forecasts, up to 50% of images acquired by operational satellites can be trashed due to excessive cloud covers, showing there is room for improvement. We propose an acquisition scheduling approach based on Deep Reinforcement Learning and experiment on a simplified environment. We find that it challenges classical methods relying on human-expert heuristic.

1 Introduction

Earth Observation (EO) systems acquire cloud-free images and deliver them to customers worldwide on a daily basis. Requests come in a variety of size and constraints, from the urgent monitoring of small areas to large area coverage. In this work we are particularly interested in the latter case, with requests covering whole countries or even continents. Depending on weather conditions, such requests may take several months to complete, even with multiple satellites.

In order to shorten the time required to fulfill requests, the mission orchestrator shall schedule acquisitions with both a short and a long-term strategy. Determining a strategy robust to an uncertain environment is a complex task, this is why current solutions mainly consist of heuristics configured by human-experts. This paper demonstrates that Reinforcement Learning (RL) might be well-suited for such a challenge. RL has proven to be of great value since these algorithms have mastered several games such as Pong on Atari 2600 (Mnih et al. 2013), Go with AlphaGo (Silver et al. 2017) and more recently Starcraft (Arulkumaran, Cully, and Togelius 2019).

2 Scheduling acquisitions for Earth observation systems

2.1 Single satellite acquisition scheduling

EO satellites carry optical instruments which are able to take acquisitions with a specific width, called swath, and a maximum length depending on the satellite models. The capacity of the satellites to take multiple images along their orbit track is related to their agility (Lemaître et al. 2002).

Due to limited swath and acquisition length, a large area must be split into tiles called meshes. For instance, considering the Pleiades satellites, covering France requires thousands of meshes. A satellite overflying an area is able to acquire a sub-part of those meshes due to its limited agility. With sun-synchronous orbit, revisit of a ground point takes days which explains the importance of mesh selection (Gleyzes, Perret, and Kubik 2012).

The satellite schedule is computed on ground by the Mission Planning Facility (MPF), where an optimization algorithm selects the top-ranked acquisitions and ensures the kinematic feasibility of the attitude maneuvers.

2.2 Interoperable EO systems scheduling for large-area coverage

The trend of EO systems is toward large constellations of heterogeneous satellites. For instance, Airbus Intelligence, operating the well-known SPOT and Pleaides satellites, will soon manage a new system of 4 satellites (Pleaides NEO). Dealing with multiple EO systems needs both human expertise and algorithms to dispatch requests over the satellites and to deliver end customers on time.

We approach the constellation scheduling by having an orchestrator responsible for request ranking towards each MPF. The orchestrator analyzes a large-scope of data (e.g., forecasts, access opportunities) to optimize the global schedule, while each MPF has a narrowed and short term vision of their single (or dual) satellite scheduling. Additionally, we focus in this paper on requests consisting in a large area (countries, continents). Such requests usually contain several hundreds

of meshes to acquire over long periods (up to several months).

The two main contributors to the overall uncertainty on the time to completion are: firstly the weather conditions at the time of acquisition, which can only be forecasted, and secondly the presence of other requests within the systems, arriving at an unknown rate.

This explains our focus on RL algorithms which have the capacity to learn new strategies, robust to uncertainties, while challenging traditional approaches.

3 Reinforcement Learning approach

In Reinforcement Learning, an agent learns how to behave through trial-and-error interactions with a dynamic environment. The actions the agent takes are decided by a *policy*, which can be seen as a function mapping information from the environment to actions. The goal of reinforcement learning algorithms is to find an optimal policy, i.e., a policy that maximizes the reward of the agent over the long-term.

Recently, deep neural networks have proven to be efficient for finding policies. Several deep-RL algorithms are actively studied to solve complex sequential decision-making problems. Among the best-known methods, one can cite value-based algorithms such as DQN, Rainbow (Hessel et al. 2018), policy-based algorithms such as REINFORCE (Sutton et al. 2000) or actor-critic methods such as A2C (Mnih et al. 2016) or PPO (Schulman et al. 2017).

3.1 Problem simplification

In order to evaluate the benefits of Reinforcement Learning, we propose a simplified environment.

We consider that all satellites have the same swath, thus the tessellation (i.e., the meshes) of the area is the same for all satellites. We also assume that each satellite can acquire at most one mesh per pass over the considered area. A satellite pass occurs when it overflies the large-area request on a given orbit. The planned mesh is validated or rejected depending on actual cloud cover observations at the time of acquisition. We do not consider uncertainties related to the load of our system, i.e., satellites are always fully available.

The area of interest (AOI) is enclosed in a rectangular box – considering a Mercator projection – containing $N_{lat} \times N_{lon}$ meshes. Since some meshes of this grid mesh may not belong to the AOI, we define $\mathcal{M} = \{m_k \ : \ 1 \leq k \leq K\}$, the set of meshes to acquire.

For each pass $t \in \mathbb{N}$, we denote by $\mathcal{M}_t \subseteq \mathcal{M}$ the subset of meshes in the AOI that can be acquired by the corresponding satellite knowing its orbit and agility.

We denote by $c_t^a(m)$ and $c_t^f(m)$ the actual and forecast cloud cover above mesh m during pass t.

3.2 Problem formulation

The given problem can be formalized as a **Markov Decision Process** (MDP) which is an intuitive and fundamental formulation for RL (Bensana et al. 1999).

An agent interacts with the environment by taking actions from a legal set of actions. The agent purpose is to acquire $\mathcal{M}$ as quickly as possible. For each step t, only one mesh can be selected. The chosen mesh is then validated or rejected depending on weather conditions.

The state space $\mathcal{S}$, the discrete action space $\mathcal{A} \subset \mathbb{N}$, the stochastic discrete-time transition function P and the reward function R define the underlying MDP: $M =< \mathcal{S}, \mathcal{A}, P, R >$.

The horizon is considered finite. Therefore, there is a finite number of discrete time steps t during an episode. Each episode comprises a maximum of $T \in \mathbb{N}^*$ steps. The state space $\mathcal{S}$ is defined as:

$$\mathcal{S} = \mathcal{S}_{status} \times \mathcal{S}_{time} \times \mathcal{S}_{passes}$$

where $\mathcal{S}_{status} = \{0,\ 1\}^{N_{lat} \times N_{lon}}$ characterizes the status of each mesh (i.e., already validated or to acquire), $\mathcal{S}_{time} \subset \mathbb{R}$ encodes the date of the current satellite pass t and $\mathcal{S}_{passes} \subset \mathbb{R}^d$ describes all pass dates, accessible meshes $\mathcal{M}_t$ and related weather forecasts.

The goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted reward over the finite horizon:

$$\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \tag{1}$$

where $0 \leq \gamma < 1$ is the discount factor and $s \in \mathcal{S}$.

3.3 Action space

At each discrete step t, the learning agent takes an action. A step corresponds to a satellite pass over the AOI and the action is to pick up a single mesh to acquire during this pass.

$$\mathcal{A} = \{0,\ 1, \ldots,\ K\}$$

We denote a_k the action selecting the mesh m_k. Note that $|\mathcal{A}| = K+1$ because there is one more "do nothing" action available for the agent.

3.4 Observation space

At a given step, the agent perceives only useful and available information about the environment. The problem is generalized to a Partially Observable Markov Decision Problem (POMDP).

The observation space O provides information about the mesh status and their validation probability for the following N_{pass} passes, including the current pass for which the agent shall select a mesh. The validation probability of a mesh depends on weather forecast accuracy, as detailed in Section 3.6. Thus, an observation is a tensor with a shape $(N_{lat},\ N_{lon},\ N_{pass} + 1)$.

The observation can be seen as a stack of $N_{lat} \times N_{lon}$ matrices. Each frame (i.e., 2D matrix) encodes information for all tiles of the grid mesh. This representation preserves spatial information and enables the use of Convolutional Neural Networks.

The validation frame encodes the status of each mesh: validated (0) or to be validated (1). We denote the validation frame space $O_{status} = \mathcal{S}_{status}$.

The validation probability frames belong to the space $O_p = [0,\ 1]^{N_{lat} \times N_{lon} \times N_{pass}}$. They encode the probability p_t to acquire and validate each mesh for each pass in chronological order from time step t. For a given mesh m and a given pass $n \in \{1, \ldots, N_{pass}\}$ at the step t:

$$p_t(m,n) = \begin{cases} 0 & \text{if } m \notin \mathcal{M}_t \\ \mathbb{P}(c^a_{t_n}(m) \leq c_{max} \mid c^f_{t_n}(m)) & \text{otherwise} \end{cases}$$

with c_{max} the total cloud cover validation threshold. $t_n = t + n - 1$ is the time related to the pass n knowing that the current time step is t.

We can now define $O = O_p \times O_{status}$

3.5 Reward

A reward is given to the agent at each step. The value of the reward depends on the status of the chosen mesh before and after this step. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ gives rewards for particular transitions between states.

$$R(s_t, a_k, s_{t+1}) = \begin{cases} 1 & \text{if } m_k \text{ is newly validated} \\ 0 & \text{otherwise} \end{cases}$$

This dense reward encourages the agent to reduce the completion time with a discount factor $\gamma < 1$ (1).

3.6 Transition function

$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,\ 1]$ denotes the transition function.

$$P(s_t, a, s_{t+1}) = \mathbb{P}(s_{t+1} | s_t, a)$$

For each transition, the current state is updated. $s^{time} \in \mathcal{S}_{time}$ takes the value of the next pass date in the chronological order. $s^{passes} \in \mathcal{S}_{passes}$ remains the same during the whole episode. $s^{status} \in \mathcal{S}_{status}$ is updated if the selected mesh is validated:

$$\mathbb{P}(s^{status}_{t+1}(m_k) = 0 \mid s^{status}_t(m_k) = 1, a_k) = p_t(m_k, n)$$

where $s^{status}_t(m_k)$ is the status of m_k at t.
This probability is computed considering the following weather model:

$$c^a_t(m) = c^f_t(m) + \chi(m)$$

$$\begin{aligned} \text{with} \quad & \chi(m) \sim \mathcal{N}(c^f_t(m), \sigma(c^f_t(m))^2) \\ \text{and} \quad & \sigma(x) = u \times x + v \end{aligned}$$

σ is a linear function computing a representative deviation between forecast and observed data.

4 Experiments

Based on the hypotheses from Section 3.1, we implement a simulator using the OpenAI Gym framework.

4.1 Scenario

To evaluate the agents, we choose mainland France as our area of interest. It is an interesting case to study because one mesh selection can have an important impact on the mission length due to the territory climate variability.

We consider 4 satellites with a common 60 km swath, implying $K = 122$ meshes for our tesselation. Each scenario begins at a random date.

We use the ERA-Interim dataset (Dee et al. 2011) which provides total cloud cover observations on a $0.5° \times 0.5°$ grid to compute the observation space. In the weather model, u is fixed to 0.1 and v to 0.2. c_{max} is set to 20% for all scenarios.

4.2 Reference algorithms

In order to benchmark the performances of our agent, we define a random agent and a heuristic that selects one mesh among $\mathcal{M}_t$ for each time step t:

- **Random** that selects the mesh randomly among accessible meshes at each pass.
- **Heuristic** that selects the mesh with the highest trade-off score between short-term and long-term probabilities p_t:

$$p_t(m,1) + \alpha \left(1 - \frac{1}{N_{pass} - 1} \sum_{n=2}^{N_{pass}} \beta^n p_t(m,n) \right)$$

  where α is the weight on future passes and β the discount factor that favors near future passes. The best performances are reached with ($\alpha = 1$, $\beta = 0.99$) for $N_{pass} = 20$.

4.3 Train and test methodology

To avoid overfitting, we use a train and test split methodology on the weather data. Training is done using data from the years 2013 and 2014, while testing is done with data from 2015.

We concentrate our experiments on the A2C algorithm which gives the best results. We train A2C agents using two observation spaces: one with a short-term vision ($N_{pass} = 1$) and one with a long-term vision ($N_{pass} = 20$). Those A2C agents are respectively named A2C-1 and A2C-20. We use the A2C implementation from the OpenAI baselines framework (Dhariwal et al. 2017) and train agents during 3×10^7 steps using 16 parallel environments ($\sim$ 30 hours using a K80 GPU and 8 vCPUs). Other hyper-parameters are set to default values.

We use a neural network architecture made of a convolution block followed by a dense block with two heads: one to estimate the state value and one to estimate the policy distribution. The convolution block contains three convolutional layers with decreasing kernel sizes (7×7, 3×3, 1×1), 128 filters per layer and ReLU activation functions. The value and policy heads are only made of a dense layer with respectively one unit and $K + 1$ units.

Figure 1 shows the mean length of the last 100 episodes as a function of the number of network weight updates for A2C-1 and A2C-20. In our environment, the length of an episode directly relates to the completion time of the area. We set a maximum number of $10 \times K$ time steps before resetting the environment

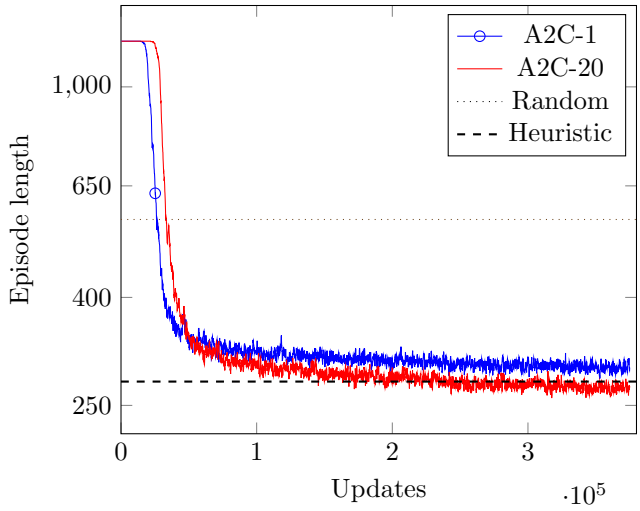

Figure 1: Episode mean length for the last 100 episodes of each training phase.

Agent	Mean	Median	Std
Random	568.8	572	110.5
Heuristic	292.7	298	56.0
A2C-1	299.3	304	58.2
A2C-20	278.5	281	55.8

Table 1: Mean, median, standard deviation of the episode lengths for the different agents.

to avoid too long episodes when the policy does not perform well. The performances of the trained agents converge close to the heuristic one. Best results are achieved with A2C-20.

During testing phase, we select days from 2015 as starting dates. For each date we assess the performances of the models and the reference algorithms. We repeat the operation using 3 different weather seeds (3×365 runs in total). Table 1 presents statistics on the episode length for the different agents. We find that for both agents the transfer on the new weather data went well.

A2C-20 still provides the best results winning the heuristic in almost 80% of the cases. It confirms the intuition that a long term strategy is necessary to optimize time-to-completion.

5 Conclusion

This paper demonstrates how Reinforcement Learning can be used in Earth Observation satellites scheduling in order to reduce the time-to-completion of large-area requests. The computed network has been trained to rank the requests and dispatch them to the satellites. In a series of simulation-based experiments, the proposed method challenges the state-of-the-art heuristics.

In future research, we aim to improve the simulation representativeness in order to pave the way for a potential industrial transfer.

References

Arulkumaran, K.; Cully, A.; and Togelius, J. 2019. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*.

Bensana, E.; Verfaillie, G.; Michelon-Edery, C.; and Bataille, N. 1999. Dealing with uncertainty when managing an earth observation satellite. In *Proc. 5th International Symposium on Artificial Intelligence, Robotic and Automation in Space (ESA SP-440)*, 205–207.

Dee, D. P.; Uppala, S. M.; Simmons, A. J.; et al. 2011. The ERA-Interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society* 137(656):553–597.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. Openai baselines. `https://github.com/openai/baselines`.

Gleyzes, A.; Perret, L.; and Kubik, P. 2012. Pleiades system architecture and main performances. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science* XXXIX-B1.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6(5):367–381.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proc. of the 33rd International Conference on Machine Learning (JMLR: W&CP vol. 48)*.

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

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.