17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.

SpaceOps-2023, ID # 220

Online Learning for Spacecraft Memory Dump Optimization

T. Cesari[a,b,c*], J. Pergoli[d,e*], M. Maestrini[e], P. Di Lizia[e]

[a] *University of Ottawa, Ottawa, Canada*
[b] *Università degli Studi di Milano, Milan, Italy*
[c] *Toulouse School of Economics, Toulouse, France*
[d] *CLC Space GmbH, Alsbach-Hähnlein, Germany*
[e] *Politecnico di Milano, Milan, Italy*
* Corresponding Authors

Abstract

In this paper, we present a real-world application of online learning with expert advice to the field of Space Operations, testing our theory on real-life data coming from the Copernicus Sentinel-6 satellite.

We show that in Spacecraft Memory Dump Optimization, a Follow-The-Leader algorithm leads to an increase in performance of over 60% when compared to traditional techniques.

Keywords: Online Learning, Mission Planning, Spacecraft Operations.

Acronyms/Abbreviations

AOS	Acquisition Of Signal
CCSDS	Consultive Committee for Space Data Systems
EO	Earth Observation
ESA	European Space Agency
FTL	Follow The Leader
LEO	Low Earth Orbit
LEOP	Launch and Early Orbit Phase
LOS	Loss Of Signal
MCC	Mission Control Center
MCS	Mission Control System
MPS	Mission Planning System
MRC	Multi Repeat Cycle
MTL	Mission Time Line
NRT	Near Real Time
NTC	Not Time Critical
STC	Short Time Critical
SRC	Single Repeat Cycle

1. Introduction

With the fast-growing number of satellites orbiting Earth, the Space Operations field has become a prominent and thriving sector. Consequently, the complexity of planning satellite operations is constantly increasing: Ground Stations must handle communication with multiple satellites simultaneously while frequently engaged in *Launch and Early Orbit Phase* (LEOP) activities; Satellite Operators need to perform routine tasks and promptly react to contingencies while checking the status of the incoming and disseminated satellite's products. These actions are costly, require time, and are remarkably prone to human errors. Despite this, Satellite Operators still carry out many of these duties by relying on their technical expertise rather than leveraging modern machine learning tools.

On the other hand, computers, hardware, and flight software are becoming more sophisticated with each passing day. At the same time, machine learning is blooming, and optimal algorithms are constantly being designed for a variety of heterogeneous problems spanning from foundational [1, 2], optimization [3, 4], cooperation [5, 6, 7], or even economics [8, 9, 10, 11]. Exploiting this growing technology means enhancing the reliability and efficiency of space operations while simultaneously avoiding human error as well as decreasing the workload on Engineers and Operators.

From an operational point of view, this entails transferring Ground Segment capabilities to the Space Segment. As outlined by [12], onboard autonomy implies:

1. Intelligent sensing: the capability to infer the system's state from environmental sensor data
2. Mission Planning and execution: the process of decomposing high-level goals into tasks that satisfy temporal and resource constraints and dispatching these tasks while monitoring and being able to react to a contingency
3. Fault management: the ability to deal with anomalies that occurred inside the system
4. Distributed decision making: effective cooperation among independent autonomous spacecraft to achieve common goals

The focal point of our paper is to identify a solution to have autonomous mission planning software onboard that can make decisions, reacting to the external environment, and dealing with anomalies while keeping operability from ground. Mission planning aims to allocate several tasks considering high-level goals, constraints, and resources. In recent years, researchers started studying this topic from different angles: using fuzzy neural networks to respond to uncertainty and proposing rescheduling [13], exploiting symmetric neural networks to improve heuristic search [14], using tabular search algorithms [15]. All these methods fall into the category of *Static Scheduling* because of the time they require to gather all the tasks. To tackle this time issue, *Dynamic Scheduling* builds a mathematical model considering five basic objects: tasks, resources, available opportunities, constraints, and objectives. In this framework, scientists have been using novel heuristic algorithms to get *close-to-optimal* results [16]; reinforcement learning based on neural network and transfer learning to solve decentralized Markov decision processes [17]; casting the problem as a stochastic dynamic Knapsack problem, and using deep reinforcement learning techniques to address it [18]. A downside of these approaches is that they require a large training dataset to output a robust decision-maker. Moreover, they often need frequent maintenance activities to be kept in line with the dynamic operational environment of a satellite.

Other proposed solutions consist of hybrid procedures, mixing autonomous onboard capabilities and ground operations. A concrete example of these kinds of systems is the FireBird mission whose scheduler exploits very simple algorithms maintaining operability from ground [19]; another one is The MER's scheduler that uses the concept of ‘onboard execution of goal-oriented mission operations’ in addition to minor support from a ground process [20].

For an *Earth Observation* (EO) Satellite, all the spacecraft activities are scheduled by the Mission-Planning Engineer weekly. Then, the tasks are uploaded to the satellite as commands during ground visibility. To improve satellite automation in modern spacecraft, the concept of position-based scheduling is exploited [21]. Hence satellites can accept three types of commands:

- *Multi Repeat Cycle* (MRC) commands, which are executed at the same orbit positions over the cycles
- *Single Repeat Cycle* (SRC) commands, which are executed at a specific orbit position for only one cycle
- *Mission Time Line* (MTL) commands, which fire at a specific time

In this paper, we present an algorithm capable of deciding when the most suitable time is to schedule the specific request to dump data from the onboard memory of a satellite (during visibility) over a ground station while keeping the system as simple as possible and *with no help from the ground*. We do this by casting our online decision-making problem into a prediction with expert advice setting [22, Section 2] for which we implement a *Follow-The-Leader* strategy [24] targeting MRC commands.

The paper is organized as follows. In Sec. 2,we give a high-level presentation of the specifics of the problem and provide a formal mathematical model for it. In Sec. 3, we introduce the Follow-The-Leader (FTL) strategy and discuss the theoretical motivations of this choice. In Sec. 4, we present the results of our experiments on real-life data, showing that implementing a simple FTL algorithm increases the performances of the satellites by more than 60%.

Figure 1: Sentinel-6 satellite. Copernicus program

2. The problem of optimizing Acquisition and Loss of Signal offsets for spacecraft memory dump

EO Satellites are designed to perform remote sensing. Most of them operate at *Low Earth Orbit* (LEO) with a period of approximately 120 minutes. Depending on the *swath* of the onboard primary mission instrument, which is the device's *areas* imaged on the surface, the satellite will complete a full Earth's coverage after a certain number of orbits. This means the spacecraft completed a cycle. Each separate orbit of a cycle is referred to as *Relative Orbit* and is repeated through different cycles. Motivation-wise, EO satellites play a fundamental role in weather forecast and natural disaster response scenarios, to name a couple. Spacecraft Operations Centres oversee distributing information to the users by commanding the satellites to perform activities over specific portions of the globe. This is done through the *Mission Control Centre* (MCC), which can receive telemetry and sending commands via the *Mission Control System* (MCS) software. Customers define the requirements and, or constraints that drive the operations performed during the lifetime of a mission. Data *timeliness* is a crucial parameter for the users' community: it is defined as the difference between the beginning of the sensing time of an instrument and the reception time of the same data on ground.

Depending on the quality level of the products, data timeliness is categorized with different labels. For Earth Observation satellites we can divide the delivered data into three main macro-categories:

- *Near Real Time* (NRT), whose timeliness is lower than, or equal to the orbit period
- *Short Time Critical* (STC), whose timeliness can be hours
- *Non Time Critical* (NTC), whose timeliness can be days even weeks

In this work, we focus on NRT data. To respect its requirements, NRT data must be dumped at every orbit without losing telemetry frames. Failing to do so results in critical data corruption, requiring another dump on the following orbit, therefore exceeding the timeliness. To protect against data corruption, a Mission Planning Engineer uploads to the onboard computer software two commands sequences for each orbit: start and stop the memory dump during visibility time. The timing of the two sequences relies on Flight Dynamic prediction of the passes over an antenna that defines the predicted *Acquisition of Signal* (AOS) and *Loss of Signal* (LOS). There are three notable categories for these pairs of events: AOS0 (resp., LOS0) is when the event occurs at the horizon, with an elevation of 0 degrees; AOSM (resp., LOSM) denotes the event taking into account the masking of other objects, such as mountains, trees, or other antennas, therefore happening after AOS0 (resp., before LOS0); AOS5 (resp., LOS5) is when the event takes place 5 degrees above the horizon, therefore it can happen before or after the masking event, depending on the specific conditions.

In Spacecraft operations, the start of the dump is always shifted by a fixed amount of time, or *offset*, with respect to the latest event between AOS5 and AOSM. The same happens for the end of the dump which is anticipated with

respect to the earliest between LOS5 and LOSM. This is done to guarantee the desired level of robustness against failures as in Figure 2.

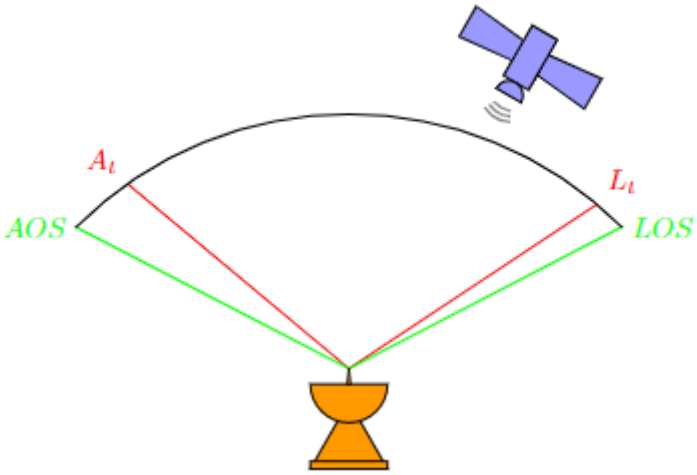

Figure 2: Pass' events timeline: AOS (resp., LOS) represents the highest events between AOS5 and AOSM (resp., LOS5 and LOSM); At and Lt represent the start and the stop of the dump which could be (and usually are) asymmetric with respect to the overall visibility.

These offsets are typically set manually and can be changed for each orbit or cycle. Usually, they are initialized with default values that consider the size of the dump and possible errors in the orbit determination and are then kept fixed, but during the early phase of a mission, things can diverge from the preparation phase, and offsets often need to be updated. In case of issues on the ground side (late acquisition and, or early loss of signal), data will be corrupted either because the satellite started dumping before locking over the antenna's signal or because the onboard computer was still downloading data while suddenly connection was cut off. The only way to prevent loss of data is to modify the offsets *ad-hoc* for each pass. This operation is not currently carried on online automatically. Therefore, to recover after an issue of this type, engineers need to manually send a request through the MCS at the next visibility opportunity. Changing these values to avoid failure is an error-prone task that puts unneeded pressure on engineers. There is no benefit in making these predictions manually. On the contrary, the nature of this problem makes it a perfect candidate for installing machine learning algorithms on onboard computers so that they can automatically learn from experience and predict online how to optimally set these offsets in order to optimize the data exchange with ground stations.

In this paper, we propose an algorithm that selects the best AOL/LOS offsets at each cycle for every orbit. Moreover, we show that implementing this approach would not only avoid the need for operators to intervene manually, but it would save a vastly larger amount of the passes when compared to the currently employed *ad-hoc* techniques.

2.1 Formal Mathematical Setting

In this section we give a brief formal introduction to the online machine learning protocol that we will use to model our offset optimization problem.

We cast our repeated interactions with the environment in the *prediction with expert advice* framework. In this formulation, each problem instance is characterized by:

1. A finite set of nonnegative times $\boldsymbol{\mathcal{O}_{AOS}}$ – the offsets from the times of Acquisition of Signal
2. A finite set of nonnegative times $\boldsymbol{\mathcal{O}_{LOS}}$ – the offsets from the times of Loss of Signal
3. A family $(p_{a,l})_{a \in \mathcal{O}_{AOS},\, l \in \mathcal{O}_{LOS}}$ of numbers in [0,1] – the probabilities of a successful dump when some AOS and LOS offsets are selected

Only the *action* set $\boldsymbol{\mathcal{O}_{AOS}} \times \boldsymbol{\mathcal{O}_{LOS}}$ is known to the learner and the best $p_{a,l}$ must be learned through data-collection. The online protocol proceeds as in Online Protocol 1

Online Protocol 1

For $\forall\, \boldsymbol{t = 1, 2, \ldots}$ **do**

The learner selects a pair of offsets

The environment draws samples $\boldsymbol{B_t(a, l) \in \{0, 1\}}$ according to Bernoulli distributions with biases $\boldsymbol{p_{a,l}}$, independently of each other and the history so far, for each pair of offsets $\boldsymbol{(a, l) \in \mathcal{O}_{AOS} \times \mathcal{O}_{LOS}}$

The learner gains the *reward* $\boldsymbol{B_t(A_t, L_t) \in \{0, 1\}}$

The learner observes as *feedback* the samples $\boldsymbol{B_t(a, l), \forall\, (a, l) \in\ \mathcal{O}_{AOS} \times \mathcal{O}_{LOS}}$

End for

Online Protocol 1

The objective of the learner is to collect as much reward as possible in any given time *horizon T*.

More precisely, denoting by α the algorithm that generates the sequence of actions $(A_1, L_1), \ldots, (A_T, L_T)$, the goal is to minimize the *regret*

$$R_T(\alpha) \coloneqq \max_{(a,l)\in \mathcal{O}_{AOS}\times\mathcal{O}_{LOS}} \mathbb{E}\left[\sum_{t=1}^{T} B_t(a,l) - \sum_{t=1}^{T} B_t(A_t, L_t)\right]$$

$$= T \max_{(a,l)\in \mathcal{O}_{AOS}\times\mathcal{O}_{LOS}} p_{a,l} - \mathbb{E}\left[\sum_{t=1}^{T} B_t(A_t, L_t)\right]$$

In words, we want the total reward accumulated by the learner to be as close as possible to that of an agent with full knowledge of the family of probabilities $\left(p_{a,l}\right)_{a\,\in\,\mathcal{O}_{AOS},\, l\,\in\,\mathcal{O}_{LOS}}$ that operates optimally by always choosing an action $(a^\star, l^\star)$ that maximizes its expected reward $p_{a^\star,l^\star}$.

3. Follow the Leader

In this section, we present the *Follow the Leader* paradigm Algorithm 1.Our choice of applying this strategy to the offset-optimization problem was initially motivated by compelling theoretical considerations (see Sec. 3.1).We then confirmed these theoretical findings empirically, showing that an implementation of a Follow the Leader algorithm does lead to dramatic performance improvements when applied to real-life scenarios (see Sec. 4).

Algorithm 1

Input: action set $\boldsymbol{\mathcal{O}_{AOS} \times \mathcal{O}_{LOS}}$, tie-breaking rule τ

For $\forall\, \boldsymbol{t = 1, 2, \ldots}$ **do**

Select action

$$\boldsymbol{(A_t, L_t) \coloneqq \operatorname*{argmax}_{(a,l)\in \mathcal{O}_{AOS}\times\mathcal{O}_{LOS}} \sum_{s=1}^{t-1} B_s(a,l)}$$

breaking ties according to τ

(With the convention that $\boldsymbol{\sum_{s=1}^{0} B_s(a,l) \coloneqq 0}$, for all $\boldsymbol{(a, l) \in\ \mathcal{O}_{AOS} \times \mathcal{O}_{LOS}}$, on time step $\boldsymbol{t = 1}$)

The learner observes as *feedback* the samples $\boldsymbol{B_t(a, l)}$, for all $\boldsymbol{(a, l) \in\ \mathcal{O}_{AOS} \times \mathcal{O}_{LOS}}$

End for

Algorithm 1

A Follow the Leader strategy maintains, during time steps *t*, an integer $\sum_{s=1}^{t-1} B_s(a, l)$ for each action (a, l), representing how well the pair of offsets (a, l) performed so far. It then simply selects an action with the best past performance, breaking ties according to the rule τ that is passed to the algorithm as an input. We highlight the role of the tie-breaking rule τ for two main reasons that are unfortunately often neglected. Firstly, from a theoretical standpoint, ties must be broken in a measurable way.* Secondly, depending on the application, some types of tie-

* Although we will not insist excessively on the mathematical formalism, it is important to know that unexpected pathologies may occur if measurability is not guaranteed. In fact, the regret might not even be well-defined without it! For more on this topic, see, e.g., [23, Section 2.4].

breaking rules are typically preferred to others. For example, in theoretical results, ties are typically broken uniformly at random (for mathematical convenience) but in practice, this is often unacceptable (because no practitioner would change a strategy that is currently working if no changes in the system are detected). We will elaborate more on this key point in Sec. 4.

3.1 Theoretical motivation

In this section, we give a concise summary of the theoretical results that motivated our choice of adopting a Follow the Leader strategy for the offset optimization problem.

A recent paper by [24, Corollary 4] shows that such a policy is optimal in the following strong sense.

Theorem 1. *For any time horizon T and any other online learning algorithm α, the FTL algorithm with ties broken uniformly at random satisfies:*

$$\sup_p R_T\,(\mathrm{FTL}) \leq \sup_p R_T\,(\alpha)$$

where the supremum is over all possible choices of probabilities $p_{a,l} \in [0,1]$*, for* $(a,l) \in \mathcal{O}_{AOS} \times \mathcal{O}_{LOS}$.

The previous result states that no algorithm can achieve better performances than FTL in general. [24, Corollary 4] goes even further, showing that in addition to the regret, the same optimality holds for expected redundancy and excess risk.

Quantitatively, it is immediate to show that FTL enjoys a *finite* regret, even when the time horizon T approaches infinity, assuming that there is at least an action (a,l) with $p_{a,l} = 1$. Indeed, in this case, for any *T*,

$$R_T(FTL) \leq 1 + \left|\left\{(a,l) \in \mathcal{O}_{AOS} \times \mathcal{O}_{LOS} : p_{a,l} \in (0,1)\right\}\right|$$

The expression on the right-hand side counts the total number of mistakes that FTL could make: on round 1, it could accidentally select an action (a,l) with expected reward zero, i.e., a pair of offsets for which it is impossible to dump successfully. All these 'bad' actions will be automatically excluded in all future rounds by definition of FTL. Then, the only way the algorithm could pay some regret on a round $t \geq 2$ is if it selected an arm (A_t, L_t) with expected reward $p_{A_t,L_t} < 1$ which happened to have given reward 1 in *all* previous rounds but returned 0 on the current one. Again, by definition of FTL, the cost of such an action could only ever be paid once and by the nature of the problem, the probability of this happening decreases exponentially with time.

These considerations give a strong theoretical foundation to the credence that FTL strategies outperform all other algorithms designed for this problem. In the following section, we will show on real data that this is indeed the case.

4. Implementation and Experimental Results

The algorithm has been evaluated by comparing the results of a schedule produced with the autonomous choice of the offsets against a real operating schedule and the real telemetry data gathered from satellites.

4.1 System Configuration

The *Mission Planning* (MP) software is a *Commercial-Off-The-Shelf* (COTS) written in C/C++ and it produces the schedule according to the schedule format from CSSDS standards [25]. The software for autonomous selection of dump sequences has been written with Python v3.7. We chose Python for the numerous support packages and modules which allow every user to personalize the tool with their own widgets, depending on their unique needs. It is also simpler than the commonly used C++ and Java, and its lower code complexity makes it easy for beginners to approach it. Simulations have been performed on a Windows 10 Pro x64 machine, processor i5-8250U 1.60 GHz, 8.00 GHz of RAM.

The real telemetry data is provided by using SCOS-2000, a generic MCS software developed by the European Space Agency (ESA). It provides the means for satellite operators to monitor and control one or more satellites. The MCS is connected to the Antenna that the satellite is downloading over, and as soon as the spacecraft and antenna signals get *on lock*, live telemetry is received and SCOS displays the first timestamp of the first telemetry frame received, doing the same when the pass is over and the last frame's timestamp is displayed. These two pieces of

information are then saved in a *csv* file together with the relative orbit number and the cycle and passed to MP software. This task has been performed by Spacecraft Controllers who were recording the time of the first and last TM frame ---but not regularly, leading to a partially filled set of data that required some pre-processing. All the test data set: telemetry, flight dynamic events, and schedule, are kind courtesy of EUMETSAT. These data recordings begin at cycle 6 of the mission because in the first cycles the satellite was not on the reference orbit, thus revolutions were not cyclic.

Regarding the flight dynamic data, these are generated by a COTS software that takes telemetry data as input and generates a file, according to CCSDS standards [26], which contains the propagation of the orbit of the satellite and its events (e.g. when the spacecraft is flying over a certain place at a certain time it save it as an event). The file is then passed to the Mission Planning software as well. After receiving all these files, the Mission Planning engineer can generate a schedule which is an *xml* file that is then sent back to MCS to generate the commands.

4.2 Scenario Configuration

The satellite considered for our tests is Copernicus Sentinel-6 and belongs to the European Earth Monitoring program. It works in an LEO orbit with an altitude of 1336 km, inclination i of 66 deg, and 127 orbits per cycle [27]. Each cycle lasts about 9.9 days. The average data dump is of 14 minutes. A default pair of values of offsets for both AOS and LOS is evaluated according to the memory dump size; in this case, we started with $A_1 = 30s$ and $L_1 = 10s$.

4.2.1 Tie-breaking rule

We break ties among leading AOS/LOS offsets by optimizing for both robustness and maximal usage of Station visibility. To do so, we first compute the smallest possible AOS and LOS offsets that could guarantee a successful dump given the historic data. Instead of selecting these offsets directly —this could be prone to failures in early rounds— we then compute a *safe* pair of offsets, that would guarantee a successful dump that starts and ends at a maximal distance from the smallest possible AOS and LOS offsets computed above.

4.3 Baselines

The FTL algorithm has been compared against data coming from real telemetry: an initial schedule has been generated with offsets set as per an operational scenario, then the schedule has been uploaded to the satellite. All the issues that occurred during the mission have been recorded together with the start and stop time of the real telemetry. This information has been compared against our FTL software.

4.4 Performance

The algorithm was run for 6 cycles of 127 orbits each. During the first 6 life cycles of the baseline mission, a total of 762 passes occurred, and 67 of them contained corrupted data due to late/early AOS/LOS. When this happens, passes need to be re-dumped the next orbit, losing timeliness. With our FTL algorithm, we were able to save more than 62% of the corrupted passes, leading to a dramatic improvement in timeliness. This was done without the need for any manual intervention. The few passes that were still lost occurred at the very early stages of the process, where the choices of *any* decision-maker are doomed by the variance. For a pictorial example of the behaviour of FTL, see Figure 3:

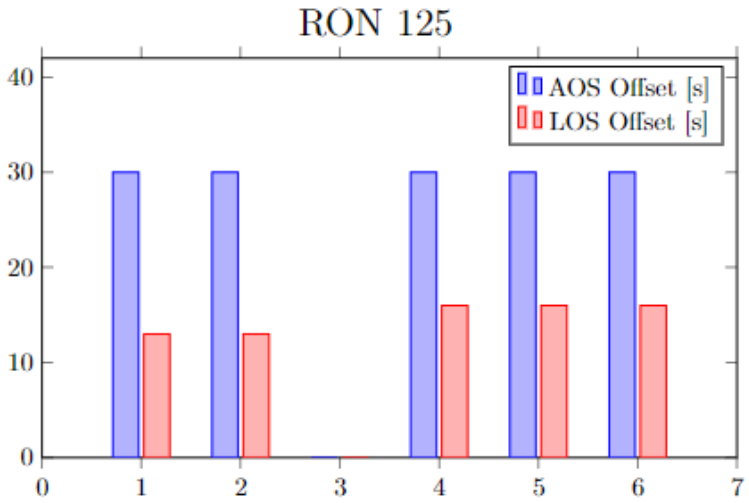

Figure 3: The LOS offset for orbit 125, initialized as 10s, is updated to 13s after time step 1, and again to 16s after time step 4. Overall, only 1 pass was lost after the initialization step. During cycle 3 no data were recorded, hence the algorithm did not run. As a reference, all 5 recorded passes

During RON 125, we received TM with gaps due to a GS masking issue. With the traditional method, after experiencing the first gap a series of investigations were starting to establish the root cause, but by the time we had to start the weekly scheduling process for the following cycle, they were not completed yet leading to another corrupted pass. Thus, this loop where we were chasing the problem rather than solving it was starting. We spent few weeks in keeping on updating these offsets values to avoid data loss. The FTL algorithm, as shown in Figure 3, would have allowed us to automatically update the offsets without manual intervention giving the engineers the required time to investigate the problem without the pressure of thinking about the best pair of values to apply at AOS/LOS. The same issue happened also during RON 101, as shown in Figure 4:

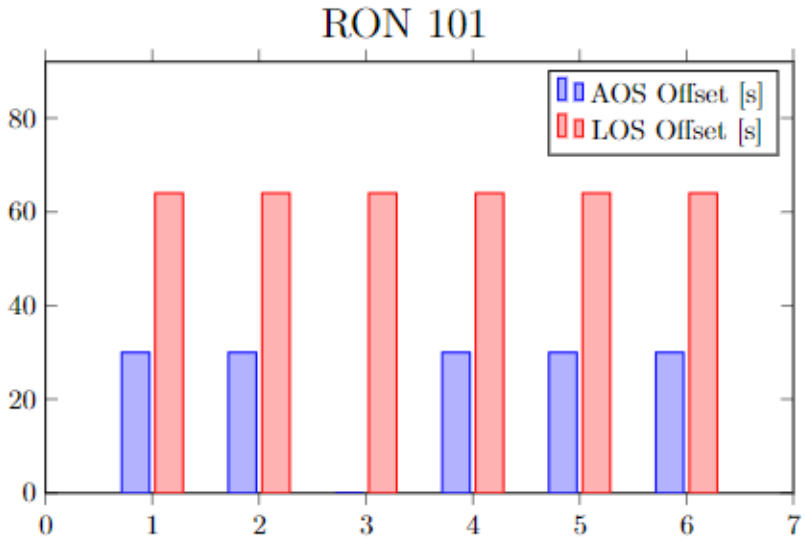

Figure 4: We initialized the LOS offset of RON to 10s, but immediately after the first cycle we experienced issues with the antenna forcing the algorithm to set the value to 64s.

What happened during the orbit number 101, highlighted also the limit of the method: the LOS offset was initially 10s, after the first cycle the algorithm set to 64s and kept following this value; in reality the offset could have also been 60s, but the algorithm stuck with the ‘leader’.

Monitoring the actual AOS and LOS is also useful in terms of anomaly detection: a value constantly changing could hide a potential issue even before exceeding the threshold. Figure 5 explains how FTL method would fit together with the already existing workflow:

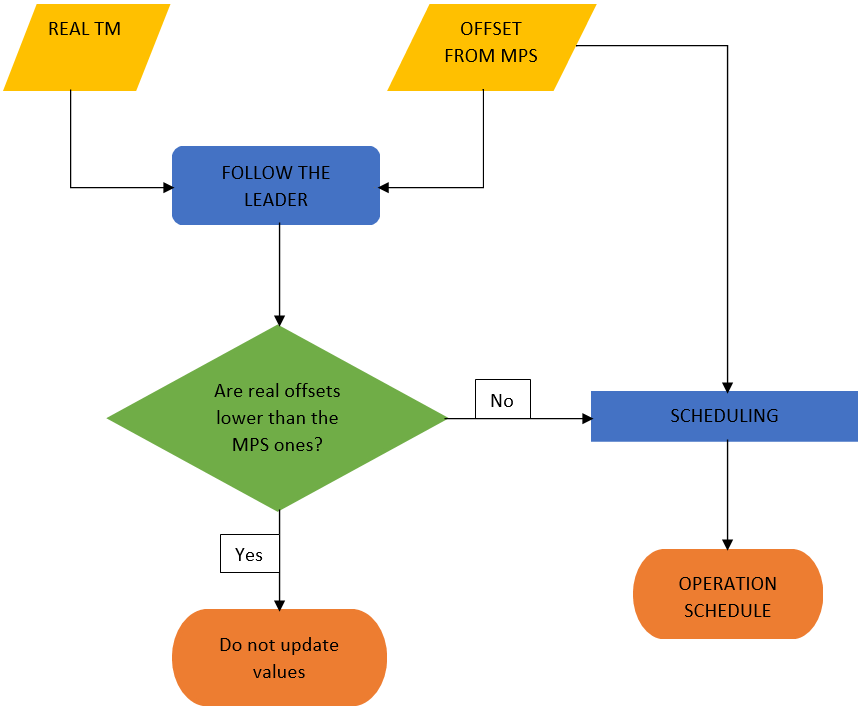

Figure 5: The data coming from the MCS and the offset values already configures in the MPS, are ingested by the algorithm after each orbit cycle. If the real TM values are below the one in MPS schedule, nothing will change; otherwise, the schedule will be updated with the new offsets.

Moreover, the algorithm would not change the value of those offset whose real telemetry is below the value set in the MPS configuration, leaving *unchanged* what is working nominally. In this way, we limit also the impact on the S/C on board software: using the position-based scheduling service, updating the command sequence to start/stop the dump requires every time a *deletion* request to remove the old command and an *addition* request for the new one, overwhelming the FCT with a lot of unnecessary sequence to check. Thus, is not convenient to insert *deletion/ addition* requests even if the system gets the full dump for specific orbits. From an operational perspective it's easier to keep track of only what is changing.

4.5 Future Works

Besides evaluating the mission performance, we also bear in mind that the software performance itself is very important: Python is an interpreted language, which is necessarily slower than a compiled one, this means that we can also speed the computation time up by using language like C++, Java or even the newest one Rust. Nevertheless, we are also already working on an upgrade of the algorithm that can reduce the calculation time of two order of magnitude. Another small limitation of FTL is that is not capable of reducing the values of the offset once it sets the *leaders*, i.e., it won't relax the conditions over a GS for a specific RON if the situation recovers. This means that if the operators want to restore the offsets values before an anomaly they must intervene manually. The modified FTL is smart enough to evaluate each single situation orbit by orbit, in case the operator wants to exploit a bigger visibility window over the stations. Therefore, the modified FTL is able of adapting through the different orbit cycles.

5. Conclusions

We presented a lightweight algorithm capable of updating the spacecraft command sequences online, autonomously, and considerably improving user requirements for data timeliness. More precisely, we manage to reach two competing goalposts simultaneously. First, *simplicity*: the Python implementation we provide consists of a relatively minute number of instructions, and intrinsically simple routines are quick to explain to engineers used to

different system practices while requiring effectively no maintenance. Second, *high performances*: our algorithm vastly outperforms the existing *ad hoc* methods, as our testing showed that switching to our FTL routine would save an enormous amount of otherwise lost data.

Acknowledgements

The authors gratefully acknowledge the support of the EUMETSAT Mission Planning team where Jonathan is working. This work started during Tommaso's Post-Doc at the Institut de Mathématiques de Toulouse, France, and benefited from the support of the project BOLD from the French national research agency (ANR). Tommaso Cesari gratefully acknowledges the support of IBM.

References

1. Cesa-Bianchi, N., Cesari, T., & Perchet, V. (2019). Dynamic Pricing with Finitely Many Unknown Valuations. *Algorithmic Learning Theory. 98*, pp. 247–273. Chicago: PMLR.
2. Bachoc, F., Cesari, T., & Gerchinovitz, S. (2021). The sample complexity of level set approximation. *International Conference on Artificial Intelligence and Statistics* (pp. 424–432). Online: PMLR.
3. Bouttier, C., Cesari, T., & Gerchinovitz, S. (2020). Regret analysis of the Piyavskii-Shubert algorithm for global Lipschitz optimization. *Regret analysis of the Piyavskii-Shubert algorithm for global Lipschitz optimization*.
4. Bachoc, F., Cesari, T., & Gerchinovitz, S. (2021). Instance-Dependent Bounds for Zeroth-order Lipschitz Optimization with Error Certificates. *Advances in Neural Information Processing Systems*, 34. Online
5. Cesa-Bianchi, N., Cesari, T., & Della Vecchia, R. (2022). *Cooperative Online Learning with Feedback Graphs*. Cooperative Online Learning with Feedback Graphs.
6. Della Vecchia, R., & Cesari, T. (2021). An efficient algorithm for cooperative semi-bandits. *Algorithmic Learning Theory* (pp. 529–552). Online: PMLR.
7. Cesa-Bianchi, N., Cesari, T., Mansour, Y., & Perchet, V. (2021). ROI Maximization in Stochastic Online Decision-Making. *Advances in Neural Information Processing Systems, 34.* Online.
8. Cesa-Bianchi, N., Cesari, T. R., Colomboni, R., Fusco, F., & Leonardi, S. (2021). A Regret Analysis of Bilateral Trade. In *Proceedings of the 22nd ACM Conference on Economics and Computation* (pp. 289–309). New York, NY, USA: Association for Computing Machinery.
9. Cesa-Bianchi, N., Cesari, T., Colomboni, R., Gentile, C., & Mansour, Y. (2021). Nonstochastic Bandits with Composite Anonymous Feedback. *Nonstochastic Bandits with Composite Anonymous Feedback*.
10. Cesa-Bianchi, N., Cesari, T., & Monteleoni, C. (2020). Cooperative online learning: Keeping your neighbors updated. *Algorithmic Learning Theory* (pp. 234–250). San: PMLR.
11. Cesa-Bianchi, N., Cesari, T., Colomboni, R., Fusco, F., & Leonardi, S. (2021). Bilateral Trade: A Regret Minimization Perspective. *Bilateral Trade: A Regret Minimization Perspective*.
12. Tipaldi, M., & Glielmo, L. (2018). A Survey on Model-Based Mission Planning and Execution for Autonomous Spacecraft. *IEEE Systems Journal, 12*, 3893-3905.
13. Li, Y., Wang, R., & Xu, M. (2014). Rescheduling of observing spacecraft using fuzzy neural network and ant colony algorithm. *Chinese Journal of Aeronautics, 27*, 678-687.
14. Liu, S., & Yang, J. (2019). A Satellite Task Planning Algorithm Based on a Symmetric Recurrent Neural Network. *Symmetry, 11*, 1–18.
15. Sarkheyli, A., Ghorbani Vaghei, B., & Bagheri, A. (2010). New tabu search heuristic in scheduling earth observation satellites. *2010 2nd International Conference on Software Technology and Engineering. 2*, pp. V2-199-V2-203. Puerto: IEEE.
16. Wang, J., Zhu, X., Yang, L. T., Zhu, J., & Ma, M. (2015). Towards dynamic real-time scheduling for multiple earth observation satellites. *Journal of Computer and System Sciences, 81*, 110-124.
17. Wang, C., Li, J., Jing, N., Wang, J., & Chen, H. (2011). A Distributed Cooperative Dynamic Task Planning Algorithm for Multiple Satellites Based on Multi-agent Hybrid Learning. *Chinese Journal of Aeronautics, 24*, 493-505.
18. Wang, H., Yang, Z., Zhou, W., & Li, D. (2019). Online scheduling of image satellites based on neural networks and deep reinforcement learning. *Chinese Journal of Aeronautics, 32*, 1011-1019.

19. Lenzen, C., Woerle, M. T., Göttfert, T., Mrowka, F., & Wickler, M. (2014). Onboard planning and scheduling autonomy within the scope of the FireBird mission. *SpaceOps 2014 Conference* (p. 1759). Pasadena: American Institute of Aeronautics and Astronautics.
20. Castano, R., Estlin, T., Anderson, R. C., Gaines, D. M., Castano, A., Bornstein, B., . . . Judd, M. (2007). Oasis: Onboard autonomous science investigation system for opportunistic rover science. *Journal of Field Robotics, 24, 379–397*
21. ESA-ESTEC. (2016). *Telemetry and telecommand packet utilization (*Vol. Requirements and Standards Division). Noordwijk, The, Netherlands: ECSS Secretariat
22. Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge university press
23. Tommaso Cesari and Roberto Colomboni. A nearest neighbor characterization of lebesgue points in metric measure spaces. Mathematical Statistics and Learning, 3(1):71–112, 2021.
24. Kotłowski, W. (2018). *On minimaxity of Follow the Leader strategy in the stochastic setting*. Theoretical Computer Science, 742, 50-65
25. for Space Data Systems, T. C. (2018). *Mission planning and scheduling* (Vol. Green Book). Washington, DC, USA: CCSDS Secretariat
26. for Space Data Systems, T. C. (2009). *Orbit data messages* (Vol. Blue Book). Washington, DC, USA: CCSDS Secretariat.
27. Wikipedia contributors. (2021). Sentinel-6 Michael Freilich — Wikipedia, The Free Encyclopedia. *Sentinel-6 Michael Freilich — Wikipedia, The Free Encyclopedia*.