J. Pergoli, T. Cesari, M. Maestrini, P. Di Lizia
Enhancing Satellite Data Integrity Through Online Learning for Memory Dump Scheduling
Journal of Spacecraft and Rockets, Vol. 63, N. 2, p. 434-445, 2026 (published online 04/11/2025)
doi:10.2514/1.a36400

Enhancing Satellite Data Integrity through Online Learning for Memory Dump Scheduling

Jonathan Pergoli*
Politecnico di Milano, Milano, Italy
CLC Space GmbH, Alsbach, Deutschland

Tommaso Cesari†
University of Ottawa, Ottawa, Canada
Italian Space Agency, Rome, Italy

Michele Maestrini‡, Pierluigi Di Lizia§
Politecnico di Milano, Milano, Italy

Managing data downlinks through memory dump scheduling in spacecraft operations is paramount to maintain data integrity and keep high mission performance. Traditional static scheduling methods lack the flexibility to adapt to random events and place a significant amount of manual workload on operators. Although machine learning techniques have shown promise, they generally require large datasets and high computational resources, both of which can be limited in practice. This paper proposes to optimize time offsets for memory dump using online learning techniques, specifically, leveraging Follow-The-Leader strategies.These lightweight sequential algorithms are based on the intuitive idea of choosing offsets with the currently best historical performance, and are known to be optimal under realistic assumptions. By integrating real-time telemetry feedback and online learning, the proposed method dynamically adjusts memory dump timings to account for variations in spacecraft operations and ground station availability, reducing the probability of data loss due to memory saturation or ground station outage. The proposed algorithm is tested using data coming from live telemetry, within the mission planning system of Sentinel-6A, demonstrating its effectiveness in optimizing memory dump by showing a remarkable improvement in data key performance index, the algorithm achieved an 86% reduction in data loss relative to the loss experienced in the real-world scenario.

*Mission Planning Operations Engineer, CLC Space GmbH, jonathan.pergoli@polimi.it.
†Assistant Professor, School of Electrical Engineering and Computer Science, tcesari@uottawa.ca.
‡Assistant Professor, Department of Aerospace Engineering, michele.maestrini@polimi.it.
§Associate Professor, Department of Aerospace Engineering, pierluigi.dilizia@polimi.it.

Nomenclature

AOS	=	Acquisition of Signal
$AOS0$	=	Acquisition of Signal 0 degrees
$AOS5$	=	Acquisition of Signal 5 degrees
$AOSM$	=	Acquisition of Signal Masking
$DORIS$	=	Doppler Orbitography and Radiopositionning Integrated by Satellite
EO	=	Earth Observation
$GNSS$	=	Global Navigation Satellite System
MCC	=	Mission Control Center
FTL	=	Follow the Leader
$FFTL$	=	Full Follow the Leader
$LEOP$	=	Launch and Early Orbit Phase
$LFTL$	=	Lite Follow the Leader
LOS	=	Loss of Signal
$LOS0$	=	Loss of Signal 0 degrees
$LOS5$	=	Loss of Signal 5 degrees
$LOSM$	=	Loss of Signal Masking
MCC	=	Mission Control Center
MCC	=	Mission Control System
MRC	=	Multi Repeat Cycle
MTL	=	Mission Time Line
NRT	=	Near Real Time
NTC	=	Non Time Critical
η	=	speed coefficient
RL	=	Reinforcement Learning
RON	=	Relative Orbit Number
SRC	=	Single Repeat Cycle
STC	=	Short Time Critical
TM	=	Telemetry

I. Introduction

With the fast-growing number of satellites orbiting Earth, the Space Operations field has become a prominent and thriving sector. As a consequence, the complexity of planning satellite activities is constantly increasing: ground stations have to handle communication with multiple satellites simultaneously while frequently engaged in *Launch and Early Orbit Phase* (LEOP) activities; Satellite Operators need to perform routine activities and promptly react to contingencies while checking the status of the incoming and disseminated satellite's products. These tasks 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.

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 [1–4]. 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 Tipaldi and Glielmo [5], on-board 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 in order to achieve common goals.

This work focuses on designing on-board autonomous mission planning software that is capable of making decisions, reacting to the external environment, and dealing with anomalies while keeping ground operability. As an initial step toward enabling automation capabilities from the ground segment to space, this work addresses the optimization of data transmission from the satellite to the antenna. This is a key aspect to improve and keep the mission performance index as high as possible because it allows to fulfill timeliness requirements especially for very high resolution data. Besides, the effort needed to re-schedule a dump of scientific data is often underestimated when a transmission fails due to antenna issues and/or limitation, keeping the stress and pressure over satellite operators at a high level. The goal of this paper is to show that it is possible to migrate the decision making to the spacecraft keeping, or even improving, the mission performance, thanks to the optimal choice of the start and stop times of data transmission over a specific antenna , for more details on the main contributions of this work, see the last paragraph of this section. Usually, the beginning and the end of the data dump is commanded from ground, and it will be executed on board at the predefined time. In addition, to avoid too frequent commanding activities, the schedule uplinked to the spacecraft typically covers several days or weeks. Even a minor issue occurring over the station where the satellite is going to download data can yield a gap in the received telemetry, which will impact in turn the integrity of scientific data. Fig. 1, from sentinel-6 altimetry level 1

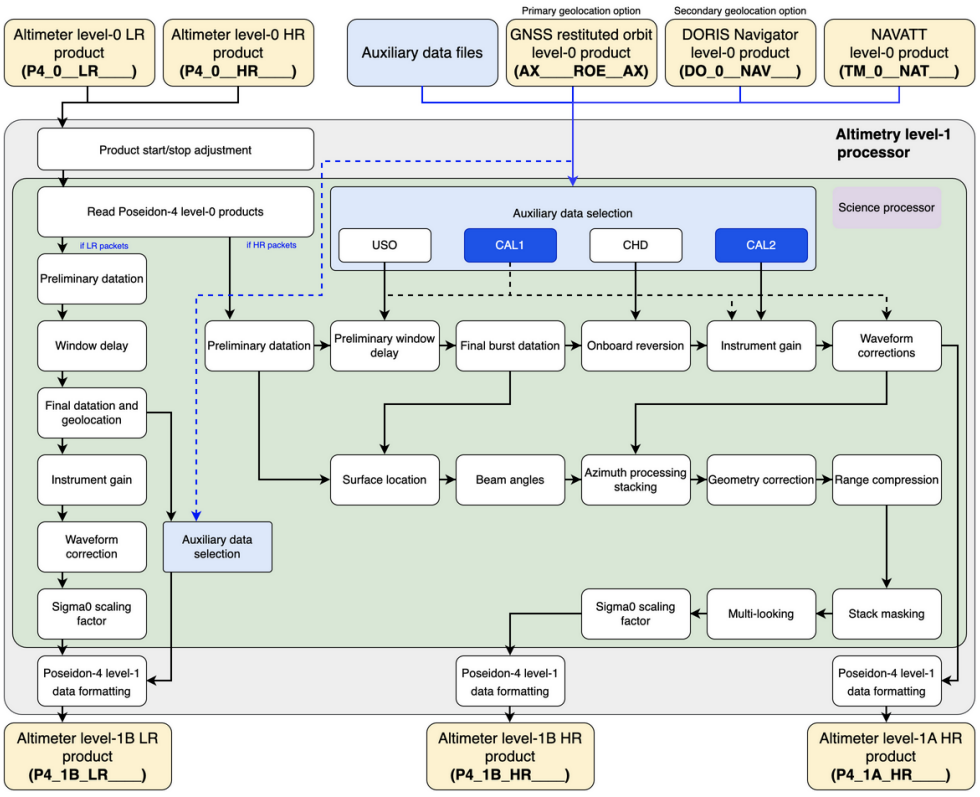


Figure 1 Flow of the production of level 1 data of the altimeter on board Sentinel-6.

data guide*, gives a clear understanding of how missing telemetry packets affect the scientific information. The figure shows a level 1 data production of Poseidon 4 on board Sentinel-6, which consists of level 0 data in *Low Rate* (LR) and *High Rate* (HR), auxiliary data and geolocation data from GNSS and DORIS. The processing chains that brings from level 0 (`P4_0`) to level 1B (`P4_1B`) LR consist of both LR and HR data, in case some level 0 data packets go missing the processing of level 1 will fail. The same can happen for the production of level 1 HR (`P4_1A`, `P4_1B`) when auxiliary of GNSS and DORIS information are missing. Thus, even small gaps in telemetry can ripple through the Level-1 altimetry processing chain, resulting in data loss, reduced accuracy, or unusable products. This is precisely why autonomous memory dump scheduling, as proposed in this paper, is essential to react promptly to antenna or link anomalies and prevent such telemetry losses before they impact data quality. Under these circumstances the on-board computer does not recalculate the new data downlink timing and all the passes over the affected station will be impacted. This means that the operation team needs to assess the problem on the antenna and re uplink new commands to avoid data corruption and maintain the required timeliness of the data. This is why a reactive and autonomous mission planning is paramount for the success of a mission. Mission planning aims at allocating several tasks considering high-level goals, constraints, and resources. For Earth Observation satellites, to maintain a high level of autonomy, the mission planning system schedules *Mission Timeline* commands. This type of commands is uplinked on regular basis, which varies depending on

*https://user.eumetsat.int/resources/user-guides/sentinel-6-altimetry-level-1-data-guide

the type of instruments and the capabilities of the mission, and then get fired once the time at which they are supposed to be executed is reached. Afterwards, they get deleted from the on-board memory requiring to be uplinked again. This applies for all EO mission. However, for satellites in *Medium* or *Low Earth Orbit*, another type of command can be exploited to overcome the limitation of regularly re-uplinking the same commands, namely the *repeated cycle* commands. These commands rely on the position of the satellite over the orbit, also said *PSO* from the French *Position sur orbit* as is defined in Fig.2, and on the fact that after a certain period of time, called *cycle*, the spacecraft fly over the same location again. In this way, once the satellite reaches the specific PSO it fires the command without deleting it because at the following cycle at the same PSO, it will the same command again. This concept allows the satellite to fly autonomously even with few commanding from ground. This concept works well in scenario where the mission

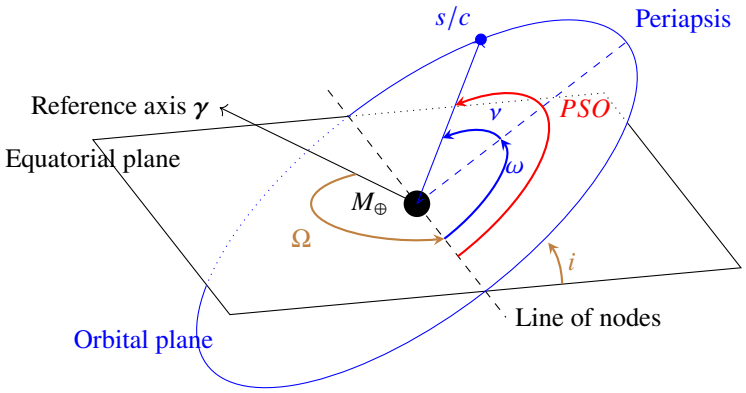


Figure 2 The *Position sur orbit* (i.e., the distance of the satellite from the line of nodes) is defined as the sum of the *true anomaly* ν and the anomaly of the pericenter ω.

profile does not change or need to change. However, modern space missions frequently evolve, making it increasingly challenging to manage recorded data across specific ground stations. As a result, innovative approaches are required to enhance onboard scheduling automation.

In recent years, researchers started studying this topic from different angles. Most of the time, the planning of a spacecraft activities is considered deterministic, therefore, it lacks of uncertainty analysis, e.g. satellite fault, satellite recovery, orbit prediction error, task cancellation, and arrival. Fuzzy neural networks can fulfill the need to cover this aspect and propose a few rescheduling solutions [6]. Using the graph coloring method to develop a new Tabu search algorithm specifically to schedule EO activities [7]. This method is not feasible in scenarios like the commissioning phase of a satellite because the neural network need to be trained on large amounts of data, which may be missing in such dynamic scenarios. Besides, the ant colony algorithm used in this paper depend on the number of steps and individuals used in the simulation which can lead to results that are difficult to interpret and to reproduce. Graph method is also used to maximize observation time or find the responsive manoeuvres [8, 9]. A schedule can also be seen as an individual and genetics algorithms can be used to evaluate the fitness of each individual [10]. All these methods fall into

the category of *Static Scheduling* because of the time required to gather all the tasks. As the name suggests, being static means they may demand hours before proposing an acceptable solution, and in space operations, especially during contingency, responsiveness is a key factor for the health of the spacecraft. To address this issue, *Dynamic Scheduling* builds a mathematical model considering 5 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 [11]. The optimality of heuristics solution can by improved by introducing *Dynamic Merging* policy to better handle resource and establishing multi-objective mathematical programming model to deal with multiple satellites dynamically [12]. These approaches perform a computation of the whole schedule each time they receive a new task, which requires a non-negligible amount of computational power that is simply not available on-board. Other proposed solutions consist of hybrid procedures, mixing autonomous on-board capabilities and ground operations. A concrete example is the system adopted for the FireBird mission, whose scheduler exploits very simple algorithms maintaining operability from ground [13]. Another example is the MER's scheduler that uses the concept of "on-board execution of goal-oriented mission operations" in addition to minor support from a ground process[14]. Considering the increasing number of missions, most of the ground segment features must be migrated toward the space segment to reduce the complexity of ground systems. An helpful hand comes from Machine Learning methods. They can improve Heuristic methods that often offer a faster alternative than the exact method, but they suffer to find the optimum when there are similar solutions, so symmetric neural networks can be use to improve heuristic search for the nominal scheduling problem [15]. A downsize of neural networks is the amount of data, power and time they need to be trained, something that in space operations is not easy to obtain: operations occur at real time and having reliable data to train the network with can take years. A major drawback of using neural networks for onboard decision-making is the significant volume of training data they typically require. In space operations, transmitting such large datasets to the spacecraft is not straightforward. Despite advances in bandwidth capabilities, uplink capacity remains limited and must be shared across many critical activities, including configuration updates, health monitoring, and command sequences. Uploading gigabytes of training data would consume a substantial portion of the available bandwidth, potentially delaying or disrupting other essential operations. Moreover, this type of data transmission must occur within relatively short visibility windows, further constraining feasibility. For these reasons, it is more practical to rely on online learning techniques, which can adapt in real time using minimal data gathered incrementally onboard, without requiring massive pre-training or large uplink volumes. Moreover, these sets need periodic maintenance activities to remain in line with the dynamic operational environment of a satellite. This issue can be solved by using **online learning** techniques by merging reinforcement learning based on neural network with transfer learning to solve decentralized Markov decision processes and avoid the failure of the historical learning caused by the randomly occurring observation requests[16]. Other works focused on the applications of reinforcement learning to high-level spacecraft planning and decision-making problems that have traditionally been the domain of optimization-focused strategies [17]. Deep Reinforcement Learning is also used to

learn spacecraft dynamic to improve Mixed-Integer Linear Programming to optimize Agile EO Satellites [18, 19] A hurdle of RL techniques is the knowledge of the environment the agent needs to explore. In order to perform actions and get precise rewards for each of them, the environment needs to be simulated correctly, otherwise the agent will be led to wrong conclusion. During delicate phase of space operations, especially where the human presence is predominant, is really difficult to create a good environment where the agent can be trained. Conversely, model-free approaches for reinforcement learning exist but require real interaction with the operative environment: an approach that is impractical in the delicate space environment due to both time constraints (typically hundreds of thousands of interactions needed) and safety concerns. Scientific users are demanding more and more data, which results in an increase in the number of real-time requests to be processed and often the lack of responsiveness of the actual systems; this need is addressed by casting the problem as a stochastic dynamic Knapsack problem and using deep reinforcement learning techniques to address it [20]. The knapsack problem is a classic combinatorial optimization problem where a set of items, each with a weight and a value, must be selected to maximize total value without exceeding a fixed capacity constraint. In satellite scheduling contexts, this analogy is used to model the selection of observation tasks or data transmissions that maximize mission utility (e.g., coverage, priority, or scientific return) within limited resources such as time, bandwidth, or energy. A downside of these approaches is that they require much computational power that is difficult to have on-board and often they cannot guarantee good performances [21].

Instead in this paper, it is proposed a simple, efficient, and explainable machine learning algorithm capable of deciding the optimal times to schedule data dumps with no help from human operators. The re-scheduling impacts only the visibility window, avoiding long computation times and the training consists only of the information of one orbit, instead of huge datasets. To fulfill this objective, the online decision-making problem is cast into a prediction with expert advice problem [22, Section 2] and implementing a Follow-The-Leader strategy [23]. The remainder of the paper is structured as follows. Section II introduces the problem context and provides a high-level overview of the memory dump scheduling challenge. Section III presents a formal mathematical formulation of the problem. In Section IV, is given the description of the Follow-the-Leader (FTL) strategy in detail and the theoretical rationale behind its adoption. Finally, Section V reports the results of experiments on real mission data, demonstrating that the proposed FTL algorithm improves satellite performance by up to 86%.

II. The problem of optimizing Acquisition/Loss of Signal offsets for spacecraft memory dump

EO Satellites are designed to perform remote sensing and play a fundamental role in a wealth of applications spanning from weather forecast to natural disaster response. Most of them are operated at *Low Earth Orbit* with a period of approximately 120 minutes. Depending on the *swath* of the on-board primary mission instrument, which is the device's *areas* imaged on the surface, the satellite will complete a full Earth's coverage, a.k.a. a cycle, after a certain number of orbits. Each separate orbit of a cycle is referred to as *Relative Orbit* and it is repeated through different cycles.

Spacecraft Operations Centers are in charge of 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 Center* (MCC) that is capable of receiving telemetry and sending commands via the *Mission Control System* software. Customers define the requirements and/or constraints that drive the operations performed during the lifetime of a mission. In particular, 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, it is possible to divide the delivered data into the following categories:

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

In this work, the focus is on NRT data. In order to satisfy its requirements, it is important that NRT data is dumped every orbit without losing telemetry frames. Failing to do so results in critical data corruption, requiring a minimum of another dump on the following orbit to recover the data, therefore, always resulting in a violation of the timeliness requirements. To protect against data corruption, a Mission Planning Engineer uploads to the on-board computer software two command sequences for each orbit: a START and a STOP command to initiate and interrupt the memory dump during the time it is in sight of a ground station. 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 (respectively, LOS0) is when the event occurs at the horizon, with an elevation of 0 degrees; AOSM (respectively, LOSM) denotes the event taking into account the masking of other objects, such as mountains, trees, or other antennas, therefore happening after AOS0 (before LOS0); AOS5 (LOS5) is when the event takes place 5 degrees above the horizon, therefore it can happen either before or after the masking event, depending of the specific conditions.

In spacecraft operations, the start of the dump d_t in a orbit t, is always shifted by an amount of time A_t, 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 the earliest between LOS5 and LOSM by some quantity L_t. Eq. 1 and 2 show how the start and the stop of the data downlink are evaluated:

$$d_{START} = \max(AOS5, AOSM) + A_t \tag{1}$$

$$d_{STOP} = \min(LOS5, LOSM) - L_t \tag{2}$$

This is done to guarantee a desired level of robustness against failures. In modern operations, satellites still rely largely on geometric predictions of AOS and LOS events based on a reference orbit. Visibility computations are typically performed using an envelope masking approach, which defines a conservative visibility window across a cluster of

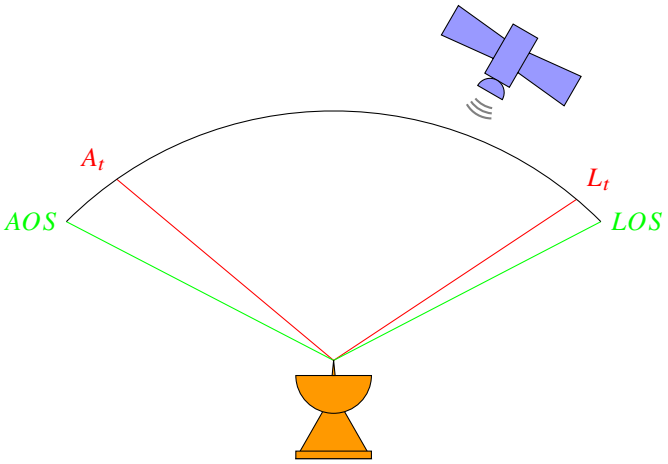


Figure 3 Pass events timeline.

nearby ground stations rather than tailoring it to each individual antenna. This method minimizes the need for frequent schedule adjustments caused by localized changes at specific antennas, which is an increasingly important consideration given the high density of spacecraft requiring frequent ground contact. Importantly, this envelope is not fixed: it may evolve over time due to changes in station performance, infrastructure upgrades, or efforts to mitigate interference and improve link quality. Any update to this masking envelope directly affects the predicted visibility window, which in turn impacts the planning and timing of data dumps. Moreover, failures in data transmission are not solely due to geometric mismatches. Ground station availability is frequently influenced by broader operational demands, such as Launch and Early Orbit Phase (LEOP) activities, routine maintenance, or contingency operations. These scenarios often force operators to change the priority of visibility slots, leading to unexpected shifts in available downlink time. In such a dynamic context, maintaining conservative AOS/LOS offsets provides a safety buffer that helps to absorb discrepancies between predicted and actual conditions. This is essential to ensure successful data transmission despite evolving ground segment configurations and mission-specific operational constraints. 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, the noise in the antenna signal, the amount of satellites the antenna needs to track because requires time changing the configuration to change from one satellite to another once the tracking is over, human error, ground stations issues, and possible errors in the orbit determination. Offsets are eventually 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, e.g., late acquisition of signal and/or an early loss of signal, data will be corrupted either because the satellite started dumping before locking over the antenna's signal or because the on-board computer was still downloading data while suddenly connection was cut off. The only way to prevent data loss 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 *Mission Control System* (MCS) at the next visibility

opportunity. Changing these values to avoid failure is a trial-and-error task that puts unneeded pressure on engineers: indeed, there is no benefit in having these predictions made manually. On the contrary, the very nature of this problem makes it a perfect candidate for adopting machine-learning techniques.

In this work, it is proposed an algorithm that automatically selects the best AOL/LOS offsets at every orbit of every cycle, and it is shown that this would not only avoid the need for operators to intervene manually but would also save a substantially larger amount of passes from failure compared to the currently employed *ad-hoc* techniques.

III. Formal Setting

In this section, it is given a brief formal introduction to the online machine learning protocol that it will be used to model the offset optimization problem: *prediction with expert advice*. This technique perfectly applies to the envisioned use case: from [22], *prediction* is the action of guessing the next action in the evolution of a phenomenon, in this case, this is a pair of AOS/LOS offests, whose performance is measured with the probability of success of a satellite dump that start and stops at the times determined by the offsets. Notably, these probabilities of success are not know ahead of time, but rather, are learned sequentially through interaction with the environment. From Sec. II, it is known that the outcomes affecting a spacecraft dump are the product of different processes that can be stochastic and/or deterministic. *Expert* algorithms do not require *a priori* knowledge of the assumptions on the data generations (although this knowledge could improve their performance in some cases). The key aspect in online learning with expert advice is to control the *regret* of learning algorithms, defined as the difference between the accumulated reward of the best fixed expert in hindsight and that of the algorithm.

In this formulation, each instance of the online learning for offset optimization problem is characterized by:

- A known finite set of nonnegative times O_{AOS} —the offsets from the times of Acquisition Of Signal
- A known finite set of nonnegative times O_{LOS} —the offsets from the times of Loss Of Signal
- An unknown family $(p_{a,l})_{a\in O_{\text{AOS}},\, l\in O_{\text{LOS}}}$ of numbers in $[0, 1]$ —the probabilities of a successful dump when AOS and LOS offsets (a, l) are selected
- An (initially) unknown family $\big(B_t(a,l)\big)_{t\in\mathbb{N},a\in O_{\text{AOS}},\, l\in O_{\text{LOS}}}$ of Bernoulli random variables such that, for all $a \in O_{\text{AOS}}$ and $l \in O_{\text{LOS}}$, $\mathbb{P}\big(B_t(a,l) = 1\big) = p_{a,l}$ and $\big(B_t(a,l)\big)_{t\in\mathbb{N}}$ is an independent family that represents whether a dump with AOS/LOS offsets (a, l) is successful at time t —$B_t(a, l) = 1$ if the dump is successful, $B_t(a, l) = 0$ otherwise.[†]

The protocol 1 unfolds as follows. The t-th time that the satellite passes over a station (for any $t = 1, 2, \dots$), the learner selects an AOS offset $A_t \in O_{\text{AOS}}$ and an LOS offset $L_t \in O_{\text{LOS}}$, and attempts to dump the data in the time window determined by A_t and L_t. At the end of this t-th visibility window, telemetry data are collected and,

[†]Time steps t represent successive cycles. The Bernoulli random variables account for randomness due to contingent conditions that can occur despite the landscape and relative positions of the satellite and station remaining the same.

Online Protocol 1

for each time step $t = 1, 2, \ldots$ **do**
The learner selects a pair of offsets $(A_t, L_t) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}$
The environment draws samples $B_t(a, l) \in \{0, 1\}$ according to Bernoulli distributions with biases $p_{a,l}$, independently of each other and the history so far, for each pair of offsets $(a, l) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}$
The learner gains the *reward* $B_t(A_t, L_t) \in \{0, 1\}$
The learner observes as *feedback* the samples $B_t(a, l)$, for all $(a, l) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}$
end for

consequently, the random variables $\big(B_t(a, l)\big)_{a \in \mathcal{O}_{\text{AOS}},\, l \in \mathcal{O}_{\text{LOS}}}$ relative to time t become known. This happens because, once it is revealed that the telemetry data was available from a time t' to a time t'', it is possible to reconstruct the counterfactual information of whether or not a dump that started and ended at times (t_a, t_l) —where t_a and t_l are the times of start/stop of the dump determined by a pair of offset (a, l)— would have been successful (regardless of the fact that (a, l) was actually selected by the algorithm and the positive or negative outcome of the dump) by simply checking if $[t_a, t_l] \subseteq [t', t'']$ and noting that $B_t(a, 1) = 1$ if and only if $[t_a, t_l] \subseteq [t', t'']$. The objective of the learner is to maximize the total number of successful dumps by a given time *horizon* T. More precisely, the goal of the learner is to determine a sequence of offsets $(A_1, L_1), \ldots, (A_T, L_T)$ that minimizes the *regret*

$$R_T \coloneqq \max_{(a,l) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}} \mathbb{E}\left[\sum_{t=1}^{T} B_t(a, l) - \sum_{t=1}^{T} B_t(A_t, L_t)\right] = \max_{(a,l) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}} T p_{a,l} - \mathbb{E}\left[\sum_{t=1}^{T} B_t(A_t, L_t)\right]. \tag{3}$$

In words, the learner strives for a total number of successful dumps that is as close as possible to that of an ideal agent with full knowledge of the family of probabilities of success $(p_{a,l})_{(a,l) \in \mathcal{O}_{\text{AOS}} \times \mathcal{O}_{\text{LOS}}}$ that operates optimally by always choosing an action $(a^\star, l^\star)$ that maximizes its probability of success $p_{a^\star, l^\star}$. Note that the expectation in Eq. (3) is not computed by the learner during execution. The regret is a theoretical measure of performance that compares the learner's cumulative loss against the best fixed action in hindsight. Since the underlying distribution $p_{a,l}$ is unknown, the learner cannot evaluate this expectation directly. Instead, our approach relies on the Follow-The-Leader strategy, which bases its decisions on empirical averages of past observations. As shown in [23], this strategy achieves minimax optimality in stochastic settings without requiring explicit knowledge of the underlying probabilities.

IV. Follow the Leader

In this section, the Follow the Leader paradigm (Algorithm 2) is presented. The choice of applying this strategy to the offset-optimization problem is motivated by compelling theoretical considerations (see Kotłowski [23, Corollary 4]) and will be further validated in Section V, where it is shown that it leads to dramatic performance improvements compared to the currently employed strategy.

A Follow the Leader strategy maintains, during time steps t, an integer $\sum_{s=1}^{t-1} B_s(a, l)$ for each pair of AOS/LOS offsets (a, l), representing how well the pair of offsets (a, l) performed so far. Then simply selects a pair of offsets

Algorithm 2 Follow the Leader (FTL)

input: action set $O_{\text{AOS}} \times O_{\text{LOS}}$, tie-breaking rule ρ
for each time step $t = 1, 2, \ldots$ **do** select action $(A_t, L_t) \coloneqq \operatorname{argmax}_{(a,l) \in O_{\text{AOS}} \times O_{\text{LOS}}} \sum_{s=1}^{t-1} B_s(a, l)$,
breaking ties according to ρ, with the convention that $\sum_{s=1}^{0} B_s(a, l) \coloneqq 0$, for all $(a, l) \in O_{\text{AOS}} \times O_{\text{LOS}}$

with the best performance so far, breaking ties according to a user-selected rule ρ that is passed to the algorithm as input. This algorithm works well in practice whenever conditions (modeled as the probabilities of successful dumps parameterized by the offsets) remain near-stationary over cycles. Whenever a large change occurs (e.g., a skyscraper is built at the boundary of the window of visibility of a ground station), historic data should not be considered reliable, and it is recommended to restart the algorithm to maintain consistency. Fig. 4 highlights the concept of Algorithm 2: after the definition of the environment $O_{\text{AOS}} \times O_{\text{LOS}}$ and the tie-breaking rules, the expert algorithm selects a pair of offsets and after the reception of the telemetry at each pass, the reward for each pair in the environment is updated and the feedback is given to update the choice for the pass in the next round. Fig. 5 gives a visual interpretation of $O_{\text{AOS}} \times O_{\text{LOS}}$.

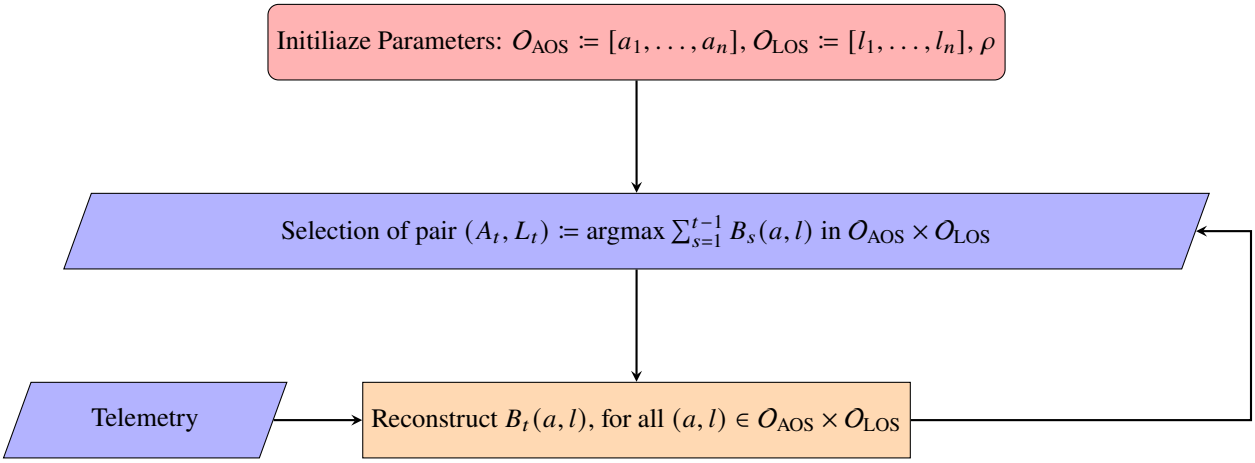


Figure 4 Follow the Leader algorithm workflow.

It is highlighted the role of the tie-breaking rule ρ for two main reasons that are unfortunately often neglected. Firstly, from a theoretical standpoint, ties have to be broken in a *measurable* way. This is needed because the regret (recall Equation (3)) is defined as expectation, and expectations are mathematically well-defined if and only if all the quantities inside the expectations are random variables (also called measurable functions). Although all "reasonable" tie-breaking rules are measurable (it is, in fact, quite hard to build non-measurable functions), it is technically possible to define non-measurable tie-breaking rules (see, e.g., [24, Section 2.4]), which would in turn break the definition of regret and lead to the loss of meaning for the performance measure of the proposed problem. This is mostly a theoretical concern as all common tie breaking rules (e.g., based on lexicographical ordering or uniformly random draws) are indeed measurable. Secondly, depending on the application, some types of tie-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 are detected). A deeper elaboration on this key point is given in Section V. In the same section (Subsection V.C), a novel "Lite" implementation of the FTL algorithm (LFTL) is also presented, which enjoys all the advantages of a FTL algorithm (see below) while requiring a minimal amount of runtime memory and computing power.

A. Theoretical Motivations

In this section, is presented a concise summary of the theoretical results that motivated the choice of adopting a Follow the Leader strategy for the offset optimization problem. A recent paper by Kotłowski [23, Corollary 4] shows that such a policy is optimal in the following strong sense.

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

$$\sup_{p} R_T(\mathrm{FTL}) \leq \sup_{p} R_T(\alpha) \tag{4}$$

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

Although this theorem is stated for a uniform tie-breaking rule, as the author states: "*Note that the proof did not require uniform tie breaking over leaders, as any distribution over leaders would work as well.*" Therefore, optimality itself holds for any way one splits probability mass among the current leaders; even for deterministic rules as the one in the Lite Follow-the-Leader (LFTL) variant introduced in Section V.C.

The previous result states that no algorithm can achieve better performances than FTL in general. Kotłowski [23, 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 when applied to the proposed problem, 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,

$$R_T(\mathrm{FTL}) \leq 1 + \left|(a,l) \in O_{\mathrm{AOS}} \times O_{\mathrm{LOS}} : \mu_{a,l}\nu_{a,l} \in (0,1)\right| \tag{5}$$

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 regret $p_{A_t,L_t} < 1$ which happened to have given reward 1 in *all* previous rounds but returned 0 on the current one. Again, by definition of FTL, such an action could only ever be played 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, it is shown on real data that this is indeed the case.

V. Implementation and Experimental Results

The algorithm has been evaluated 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 a satellite.

A. System Configuration

The software prototype is developed with Python 3.7. Python's syntax is simpler and more intuitive, allowing for faster implementation and iteration, which is crucial during the early stages of software development when speed and flexibility are key. Additionally, Python's extensive libraries greatly accelerates the prototyping process. While programs like C++ or Java offer performance advantages, Python's flexibility and development speed make it an ideal choice for quickly building and testing. 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. The flight dynamic software propagates the orbit of the satellite and generate events, e.g. when the spacecraft is flying over a certain place, with a timestamp, the position on the orbit and the duration of each event. The events are then passed to the Mission Planning Software. The MPS processes all the info and produces the updated schedule. The data flow for this testing is shown in Fig. 6. At each pass, after starting the contact with the satellite, the MCS receives the telemetry frames and sends the information about the beginning and ending of contact to the MPS. MPS compares the information with the prediction from the orbit propagator and autonomously decides which is giving a more stringent window of visibility in order to dump the data while minimizing the risk of losing any of it. The MPS does the same process for each orbit of the next cycle. Afterwards, it updates the new commanding sequences, if necessary, and sends back the new schedule to the MCS to be then uplinked on board the satellite.

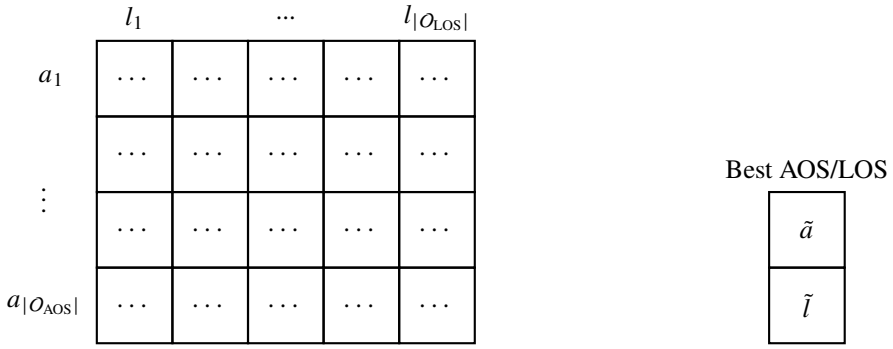
B. Scenario Configuration

The satellite considered for the proposed tests is Copernicus Sentinel-6 and belongs to the European Earth Monitoring program. It operates in a LEO orbit with an altitude of 1336 km, inclination i of 66 deg and 127 orbits per cycle [25]. Each cycle lasts about 9.9 days and the average data dump lasts 14 minutes. A default pair of memory dump offsets for both AOS and LOS is evaluated according to the memory dump size; in this case, the scheduling is initialized with $A_1 = 30s$ and $L_1 = 10s$. The choice of default values for AOS and LOS offsets is driven primarily by two factors: the geometry of the orbit and the characteristics of the receiving antenna. The lower bounds of these ranges are carefully selected to account for potential signal acquisition issues near the edges of the visibility window. Specifically, the

geometry of the orbit, including elevation angles and local horizon masking caused by terrain or infrastructure around the ground station, can delay signal acquisition beyond the nominal AOS or anticipate the LOS. Likewise, the antenna's performance, including the signal-to-noise ratio (SNR) and the time needed for stable signal lock, introduces additional uncertainty. Starting the data dump too close to the predicted AOS can result in early transmission before a reliable link is established, leading to partial or total data loss. Similar considerations apply to LOS: stopping too late risks losing telemetry as the signal weakens or is lost due to low elevation or masking. Therefore, these default offset ranges serve as conservative safety margins designed to absorb variations in real-world conditions, ensuring robust operation across a wide range of passes.

C. Vanilla and Lite Implementations

A formal presentation of the FTL strategy in its vanilla form, which maintains the cumulative reward for all actions and selects the current leader, is provided in [22, Chapter 2]. In a vanilla implementation of the FTL algorithm, the offsets to be chosen are selected from an array with a minimum of 30 to a maximum of 150 seconds for AOS, and a minimum of 10 to a maximum of 150 seconds for LOS. Greater values are not practically feasible as they would make it impossible to cover the entire orbit: note that these parameters depend on the specific type of mission. The number of orbits and cycles can vary according to the specific mission and are considered an input of the software. An issue of a vanilla FTL implementation is that keeping up-to-date a large matrix of AOS/LOS *indices* (the past numbers of successes of each pair) can be memory and time-consuming. This problem is solved with a lighter version (LFTL) that maintains only a *sparse* set of data and is able to make a decision in *constant time* based on them. The reduction in overhead is particularly important since the aim of the whole algorithm is to run on-board where resources are limited. Even when resources are abundant, the lite version has no downsides compared to the vanilla implementation, allowing to save some memory at no additional cost. Therefore, there is no reason to prefer the latter to the former.



(a) Memory consumed by vanilla version. **(b) Memory consumed by lite version.**

Figure 5 Visual representation of the memory occupied by maintaining the historical performance of all AOS-LOS offset pairs ($|O_{\text{AOS}}| \times |O_{\text{LOS}}|$ matrix) versus simply keeping track of the current-best pair of indices (2 indices).

Tie-breaking rule. The ties are broken among optimal AOS/LOS offsets by balancing the two competing goals of robustness and maximal usage of station visibility. To do so, firstly the smallest possible AOS and LOS offsets that could guarantee a successful dump given the historical data is computed. Instead of selecting these offsets directly (which could be prone to failures in early rounds), then 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 is evaluated. In early rounds, such safe offsets were selected, but they eventually converge to the largest possible size of time visibility. This time expansion is regulated with a parameter η, which is also a software input. This choice of tie-breaking rule guarantees three important *desiderata*:

1) Firstly, choices are deterministic, so that no changes would occur unless new data mandates it.
2) Secondly, by having looser offsets at the beginning, it is possible to avoid losing early passes by not overfitting to the initial limited data.
3) Thirdly, over time, convergence to the maximum usage of the ground station visibility window is reached.

Implementing the sparse memory consumption described above and selecting a tie-breaking rule according to the three desiderata, Algorithm 2 becomes Lite Follow the Leader (LFTL) (Algorithm 3). Note that, under the minimal assumption that there exists a time interval (possibly very small) inside which dumps are always successful (i.e., that there exists a pair (a, l) such that $p_{a,l} = 1$), LFTL is indeed a Follow-The-Leader (or FTL) algorithm. This follows from two considerations. First, given the binary nature of the feedback, the only offsets among which FTL chooses the pair to use are those corresponding to a start/stop of the dump for which no dumps have been lost in the past. Second, by the structure of the problem, if the telemetry at a time step t begins at a time τ_t^{AOS} and ends at a time τ_t^{LOS}, then the only candidates being chosen at time $t + 1$ are the offsets whose corresponding starting/stopping times fall in the interval $[\tau_t^{\text{AOS}}, \tau_t^{\text{LOS}}]$. Therefore, there is no need to track the performance of *all* individual pairs of offsets over time (like a vanilla FTL would do), but it is sufficient to maintain the highest τ_t^{AOS} and lowest τ_t^{LOS} observed so far to be able to reconstruct the interval of all historically-optimal offsets. The algorithm then break ties among this set of pairs of optimal points by selecting the endpoints of an interval that grows exponentially fast to $[\max_{s\le t} \tau_s^{\text{AOS}}, \min_{s\le t} \tau_t^{\text{LOS}}]$, in a way regulated by a parameter η. As can be seen in Algorithm 3, small values of η yield large dump windows early on, while larger values are more conservative.

D. Baselines

The FTL algorithm has been compared with data coming from real telemetry. In real operations, the spacecraft schedule is created and uplinked on a weekly basis. When, for any reasons, a dump becomes corrupted, this is requested manually on the next pass, but the schedule for the next week will still have the same offsets for two reasons: firstly, these type of commands are MRC and is preferable to not delete them from the on-board queue to re-uplink them unless it is strictly required; secondly, the offsets are changed manually at schedule generation level, therefore it is time

Algorithm 3 Lite Follow the Leader (LFTL)

input: expansion rate $\eta \in (0, 1)$, τ_0^{AOS} and τ_0^{LOS} are the first instants when telemetry is received.
initialization: let $l_0 \coloneqq \tau_0^{\text{AOS}} \coloneqq \max\{\text{AOS5}, \text{AOSM}\} + \min O_{\text{AOS}}$, $r_0 \coloneqq \tau_0^{\text{LOS}} \coloneqq \min\{\text{LOS5}, \text{LOSM}\} - \min O_{\text{LOS}}$
for each time step $t = 1, 2, \ldots$ **do**
  compute the time d_t needed for the current dump,
  set $l_t \coloneqq \max\{l_{t-1}, \tau_{t-1}^{\text{LOS}}\}$, and set $r_t \coloneqq \max\{r_{t-1}, \tau_{t-1}^{\text{AOS}}\}$
  begin the dump at time $\left\lceil l_t + \eta^{t-1} \cdot (r_t - d_t - l_t)/2 \right\rceil$
  end the dump at time $\left\lceil r_t - \eta^{t-1} \cdot (r_t - d_t - l_t)/2 \right\rceil$
  record the moments τ_t^{AOS} and τ_t^{LOS} when the telemetry began and ended this time step
end for

consuming to change them unless the issue becomes recurrent. Hence, many passes are lost before confirming that the problem belongs to the antenna and not to the satellite. This could have been enhanced if, rather than relying on MRC requests, the onboard computer had utilized MTL to automatically determine the start and stop of the transmission, as achieved by the proposed algorithm. In this case, it is initially generated a schedule exactly reflecting the one that was on-board, afterward the real telemetry coming from the MCS is monitored to run the algorithm taking information from it to re-adjust automatically the offsets and change the start and stop of the commands of the dump of the satellite. All the issues that occurred during the mission have been recorded together with the start and stop times of the real telemetry. This information has been compared to the proposed FTL software. A comparison was also made between the vanilla and lite FTL to show that the performance does not worsen, but the computational time for each orbit improves significantly. This is a crucial difference because satellite resources are finite, and one must ensure that the mission planning software is not draining them, preventing the satellite from performing its nominal activities. It is important to emphasize that other machine learning algorithms were deemed unsuitable for the outlined reasons from the outset of the development phase.. Traditional models usually require a very large set of data which at the beginning of the mission lifetime is difficult to obtain. Moreover, at the early stage, the understanding of the mission environment can be limited. Thus, the learning phase of traditional model may not lead to the required accuracy level.

E. Performances

The algorithm was run for 6 cycles of 127 orbits each. During the first 6 life cycles of the mission, a total of 762 passes occurred, and 67 of them contained corrupted data due to late/early AOS/LOS. Considering that a LEO satellite has on average 13 passes per day and that the losses were around the 9%, it means that almost one orbit out of ten was corrupted which means that more than one pass per day needed to be re-scheduled. When this happens, passes need to be re-dumped in the next orbit, losing timeliness. As explained in section V.D, before changing them on the satellite, operators had to ensure what was the issue with the antenna since updating them is a manual activity, hence during the analysis many other passes were requested manually due to the wrong values of the offsets. Fig. 7 shows an example of how the algorithms works: for RON 125 at cycle 0 operators made the first schedule considering to start the data

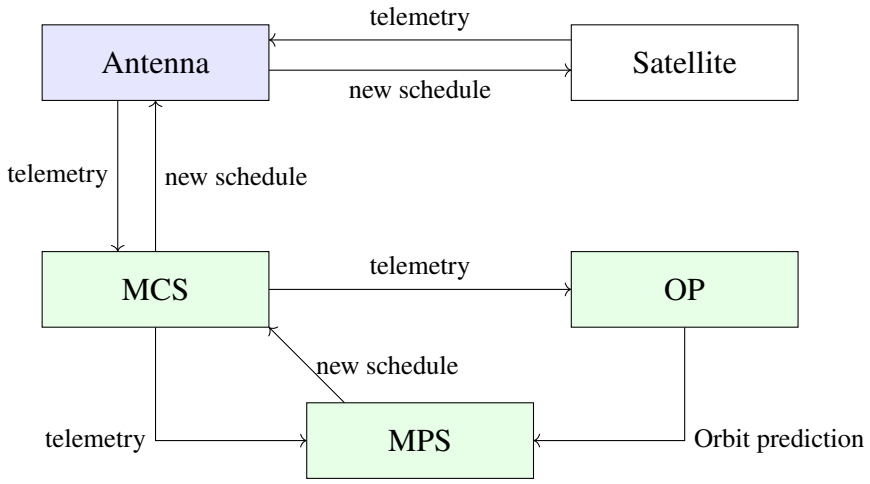


Figure 6 Data exchange between Antenna, Satellite, MCS, MPS, and Orbit Propagator for generating a new mission schedule.

transmission with an offset of 30s while for the stop is 10s. The expert made no predictions at beginning because it had to wait for TM to come first. Once the software observed from real telemetry that there was an early LOS, from TM it arrived 12s after the planned LOS while the command is set to be fired after 10s, so the expert made its prediction following 3 giving an offset of 118s at cycle 1. After observing the first cycle, where the offset from TM would have been 11s, way lower than 118s so the pass was successful, the expert adjusted its forecast to be equal to the TM one since $\eta = 0$. Unfortunately, at cycle 2 the situation worsened again and since the expert suggested to use 11s while the TM showed 12s, the pass was lost, but the prediction changed from 11s to 12s. From cycle 3, the situation did not change because the real offset increased again to 15s, but luckily it remained constant for the following cycle. . Whenever the offsets predicted by the expert is greater or equal to the one coming from TM, it means the pass was successful. In real operations, since the default offsets were always 10s, and the TM was reporting that they needed to be higher, all the passes were lost, while with the LFTL the system would have been able to recover 2 of them even with $\eta = 0$, meaning that the algorithm was just following the environment behavior. In Fig. 8, increasing $\eta = 0.2$ improved the behavior of the prediction, saving one pass more in each orbit.

With the proposed FTL algorithm, the system would have been able to save more than 86% 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 in the very early stages of the process, where the choices of *any* decision-maker are doomed by the variance. To prove the results, this paper presents the schedules of the first 6 cycles and checks that the algorithm was scheduling the start and stop of the data transmission at the same time or after the first TM packet received and at the same time or before the last TM received as shown in Fig. 9

As mentioned earlier, the performance of the FTL algorithm depends on the parameter η that regulates how much the decision-maker is willing to push offsets close to the boundary of the unsafe region (see Fig. 10). This parameter set

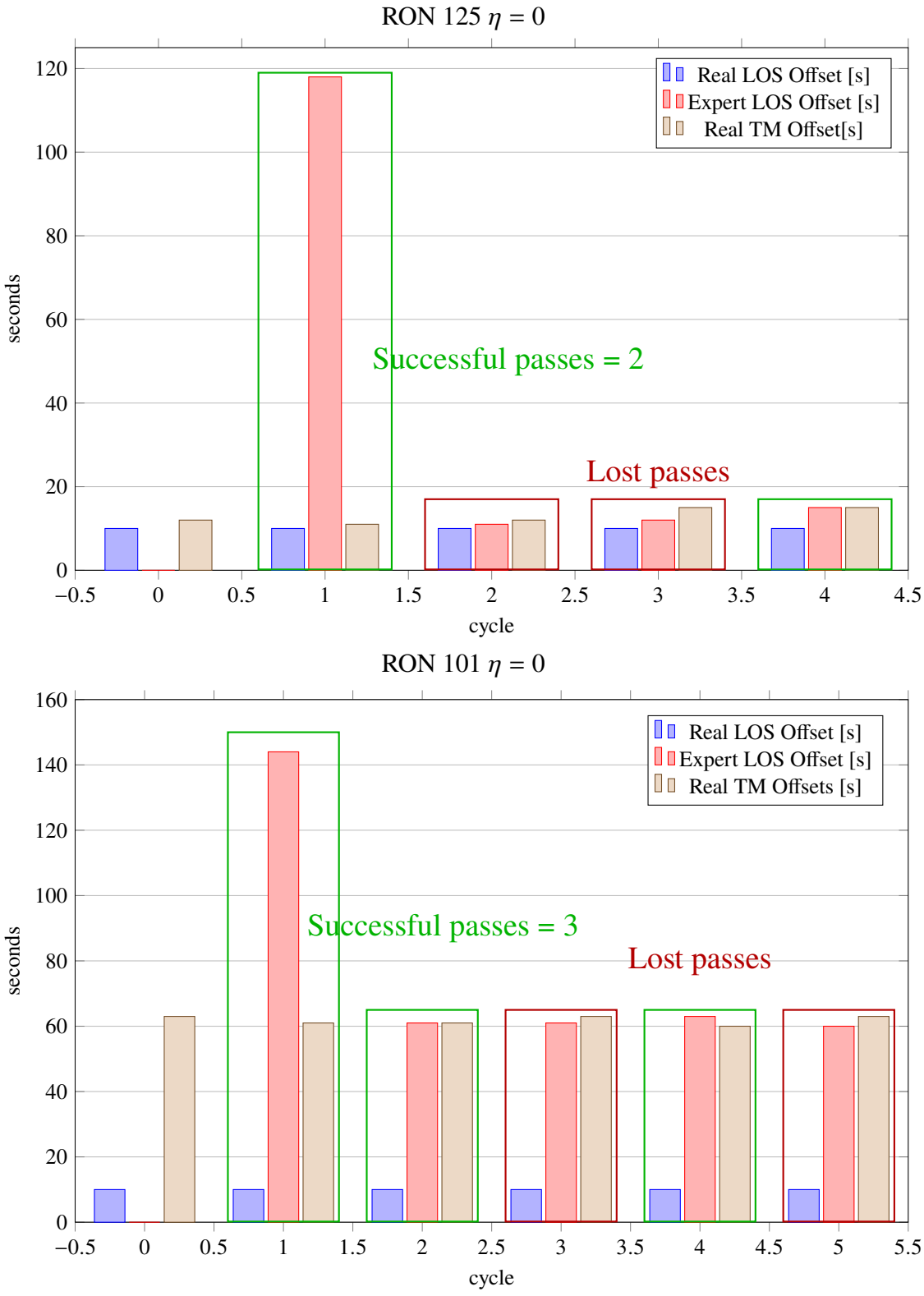


Figure 7 Example of expert reasoning.

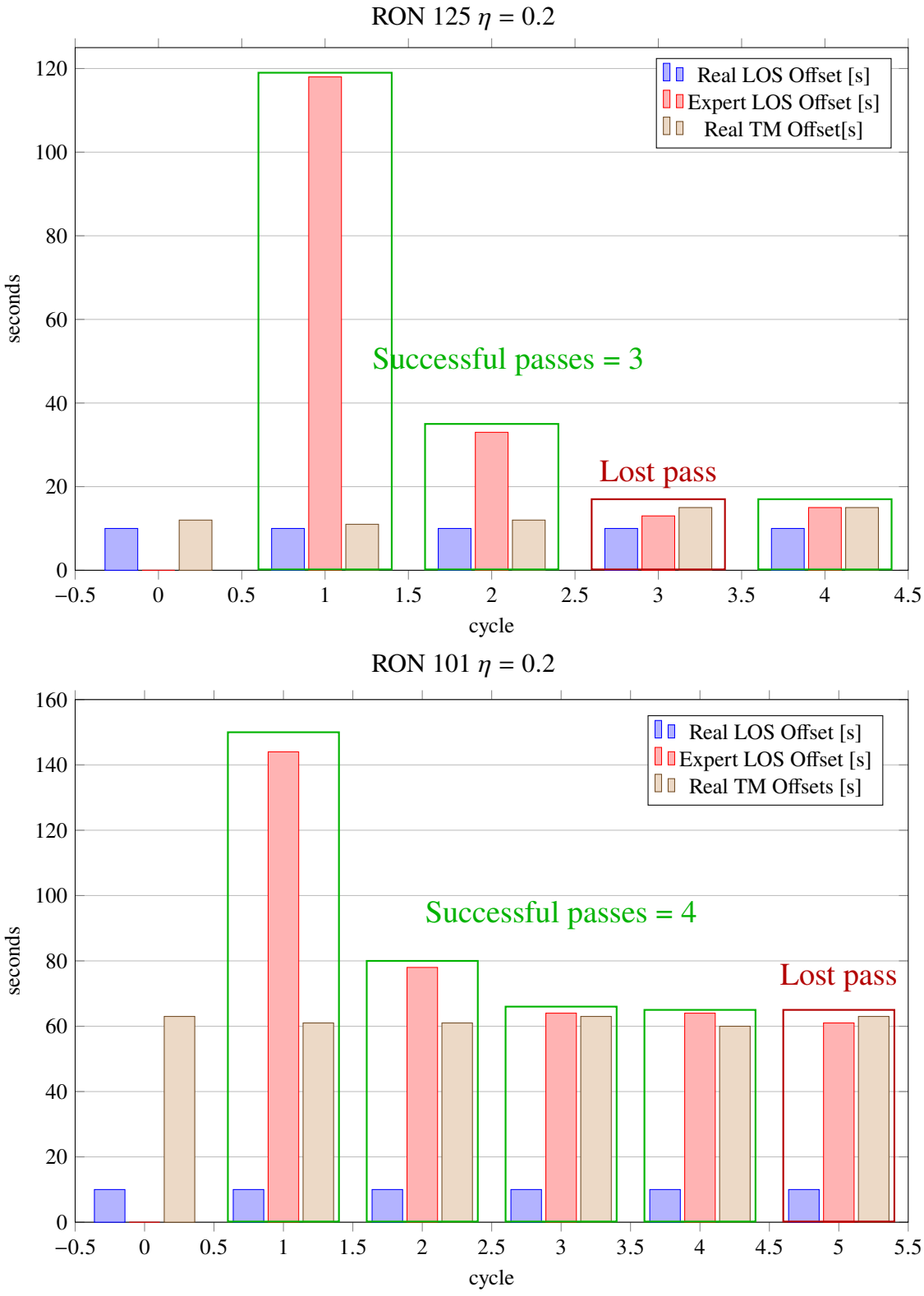
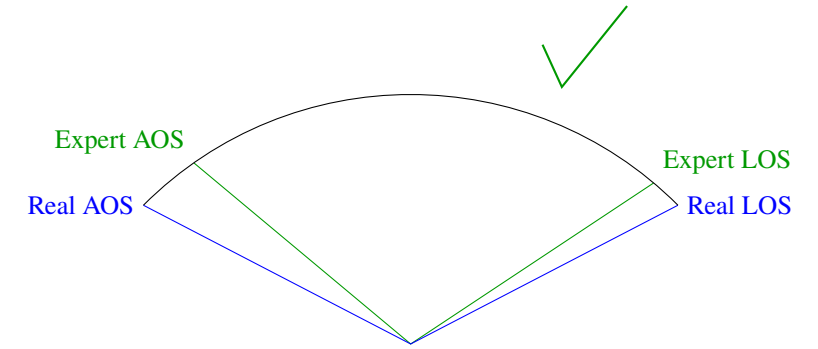
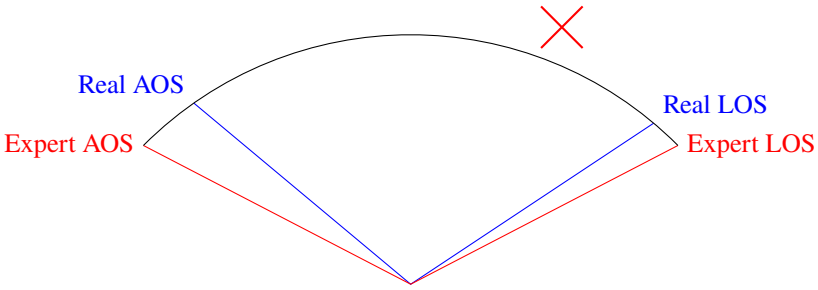


Figure 8 Effect of a greater η on the expert reasoning.



(a) Correct expert estimation.



(b) Wrong expert estimation.

Figure 9 Two different predictions from the algorithm: in (a) the expert correctly schedules the request to happen after good link between the satellite and the antenna; in (b) the expert made a poor choice.

the behavior of the learner to be more robust or more dynamic: the lower its value is, the faster it will change offset from cycle to another reaching convergence to configured values very early in time. Conversely, the higher this value is, the more robust it will make the learner to unexpected malfunctioning, but it will slow down convergence to the the desired values. As a consequence, it was decided to leave this parameter as a software configuration variable to allow the operator to retain control on the algorithm. This choice relies on the fact that the human supervision in the loop is still strongly required in certain space operations and removing it completely can impair adoption of new technology.

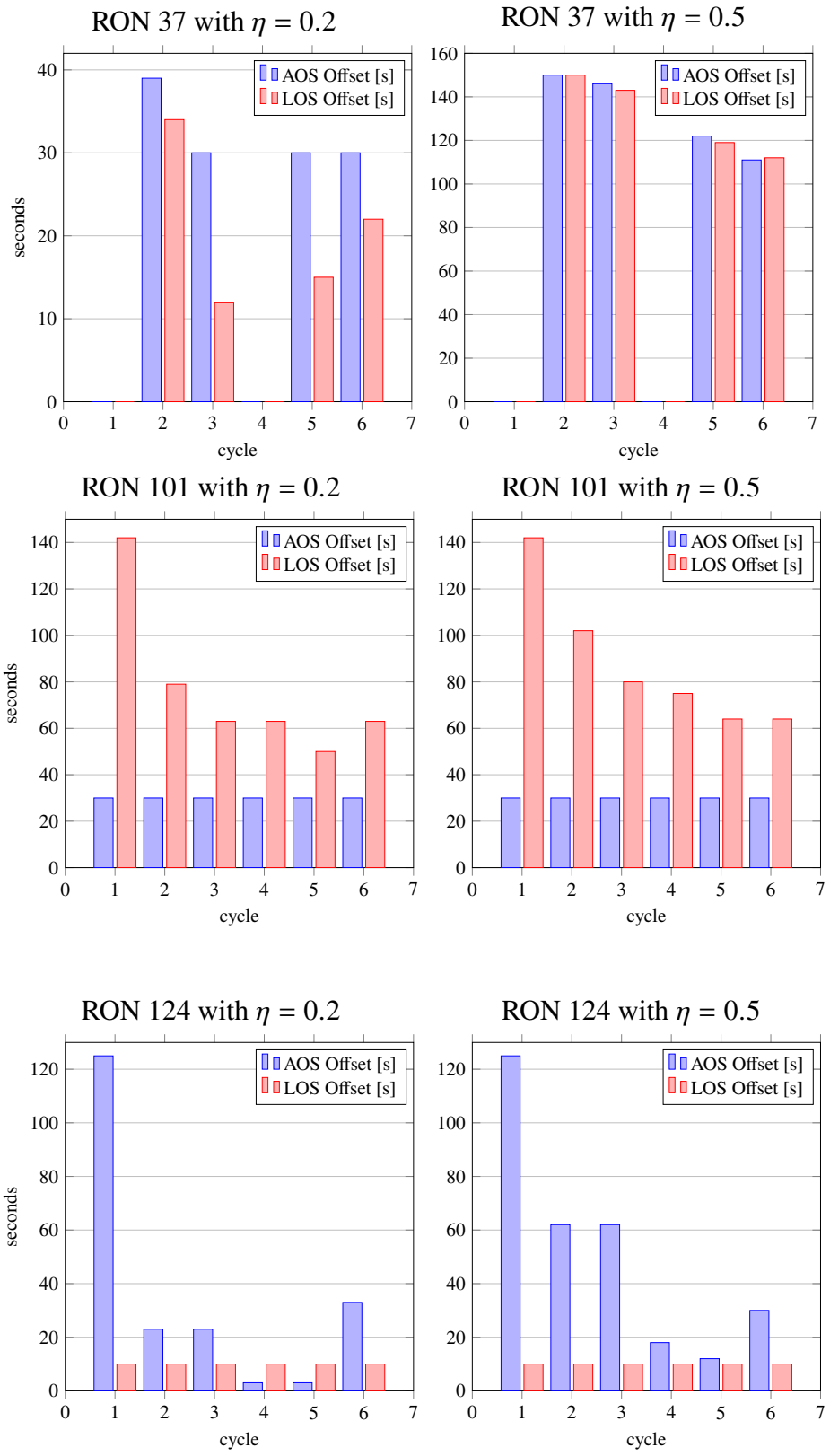


Figure 10 Impact of η on the rate of success.

In Table 1, vanilla and lite version with respect to a real scenario are compared.

Table 1 Percentage of succesful passes

Algorithm	Lost Passes	Saved Passes	%	η
Real Life	67	0	0	
VFTL	25	42	62	N/A
LFTL	37	30	44	0
LFTL	14	53	79	0.3
LFTL	10	57	85	0.5
LFTL	9	58	86	0.8

It must be highlighted that even though higher values of η *initially* trade-off station-visibility time in favor of robustness, the algorithm converges quickly to the minimum values for the offset, allowing significantly more real-time telemetry as shown in Fig. 11.

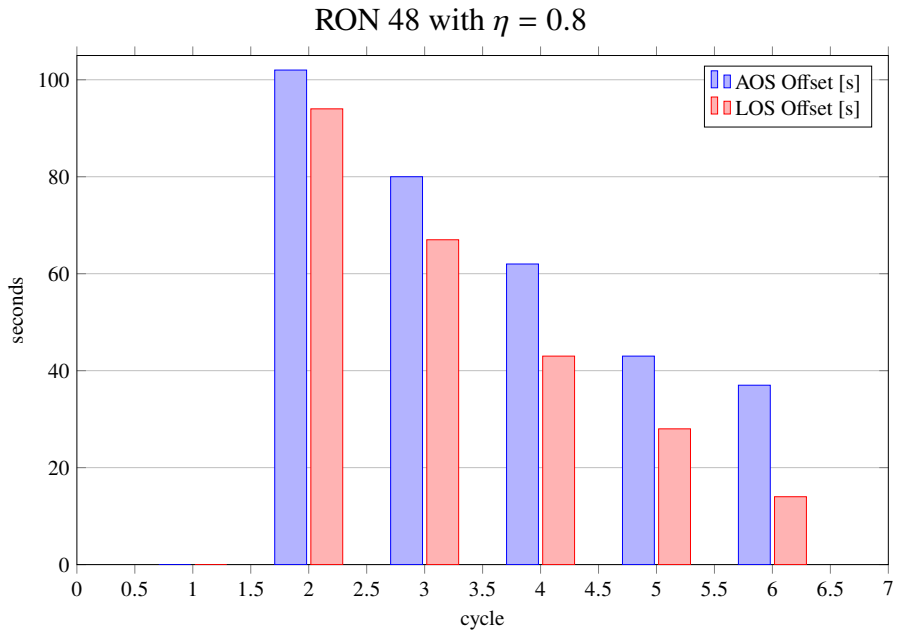


Figure 11 Convergence of the algorithm with η = 0.8

It is also important to underline that LFTL reduces the calculation time by *two orders of magnitude*: VFTL takes $1.345s$ to find the right offset for one single orbit while LFTL takes only $0.02s$ (see Table 2). If needed, the computation time can be reduced even further by using, for example, Cython, which brings the speed of C++ to Python.

VI. Conclusions

In this paper, it is presented an algorithm capable of updating the spacecraft command sequences online, without manual intervention, and considerably improving user requirements for data timeliness. More precisely, the algorithm

Table 2 Computational time difference between VFTL and LFTL: the analysis has been performed for a single orbit and for a single cycle.

Algorithm	For orbit	For cycle
FFTL	$1.345s$	$85.794s$
LFTL	$0.02s$	$1.71s$

managed to achieve four important goals simultaneously. First, *simplicity*: The Python implementation consists of a relatively minute number of instructions. Intrinsically simple routines require reduced maintenance effort (i.e., low code complexity) and are quick to explain to engineers coming into new system practices. This is a very important point from the perspective of spacecraft operators for whom clarity and comprehensibility in any point of the mission is a key aspect. Secondly, *reduced computational complexity*: the proposed algorithm requires only a constant number of elementary operations per time step and performs each update in milliseconds, making it perfect for on-the-fly machine learning. Third, *limited memory requirements*: the proposed lite LFTL implementation reduces the space requirement of the vanilla VFTL significantly, making it essentially independent of the size of the action set. This property guarantees that running the algorithm does not impose any measurable limitations on the amount of data the satellite can collect. These latter points are crucial for on-board implementation since resources must be used parsimoniously. Last but not least, *high performances*: the proposed algorithm vastly outperforms the existing *ad hoc* methods. As illustrated in section V.E, the performed tests showed that switching to FTL would save an enormous amount of otherwise lost data compared to the manual operations that are currently in place.

Acknowledgments

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). TC also gratefully acknowledges the support of the University of Ottawa through grant GR002837 (Start-Up Funds) and that of the Natural Sciences and Engineering Research Council of Canada (NSERC) through grants RGPIN-2023-03688 (Discovery Grants Program) and DGECR2023-00208 (Discovery Grants Program, DGECR - Discovery Launch Supplement).

References

[1] Peng, H., and Bai, X., "Artificial Neural Network–Based Machine Learning Approach to Improve Orbit Prediction Accuracy," *Journal of Spacecraft and Rockets*, Vol. 55, No. 5, 2018, pp. 1248–1260. https://doi.org/10.2514/1.A34171, URL https://doi.org/10.2514/1.A34171.

[2] "Introduction to Advancements in Machine Learning with Aerospace Applications Virtual Collection," *Journal of Spacecraft*

and Rockets, Vol. 61, No. 6, 2024, pp. 1430–1430. https://doi.org/10.2514/1.A36245, URL https://arc.aiaa.org/doi/abs/10.2514/1.A36245.

[3] D'Ambrosio, A., Carbone, A., and Curti, F., "Optimal Maneuvers Around Binary Asteroids Using Particle Swarm Optimization and Machine Learning," *Journal of Spacecraft and Rockets*, Vol. 60, No. 5, 2023, pp. 1458–1472. https://doi.org/10.2514/1.A35317, URL https://doi.org/10.2514/1.A35317.

[4] Fereoli, G., Schaub, H., and Di Lizia, P., "Meta-Reinforcement Learning for Spacecraft Proximity Operations Guidance and Control in Cislunar Space," *Journal of Spacecraft and Rockets*, Vol. 0, No. 0, 0, pp. 1–13. https://doi.org/10.2514/1.A36100, URL https://doi.org/10.2514/1.A36100.

[5] Tipaldi, M., and Glielmo, L., "A Survey on Model-Based Mission Planning and Execution for Autonomous Spacecraft," *IEEE Systems Journal*, Vol. 12, No. 4, 2018, pp. 3893–3905.

[6] Li, Y., Wang, R., and Xu, M., "Rescheduling of observing spacecraft using fuzzy neural network and ant colony algorithm," *Chinese Journal of Aeronautics*, Vol. 27, No. 3, 2014, pp. 678–687.

[7] Sarkheyli, A., Ghorbani Vaghei, B., and Bagheri, A., "New tabu search heuristic in scheduling earth observation satellites," *2010 2nd International Conference on Software Technology and Engineering*, Vol. 2, IEEE, Puerto Rico, USA, 2010, pp. V2–199–V2–203.

[8] Burgon, R., Roberts, P. C. E., Roberts, J. A., and Ankersen, F., "Science Operations Planning Optimization for Spacecraft Formation Flying Maneuvers," *Journal of Spacecraft and Rockets*, Vol. 46, No. 3, 2009, pp. 634–644. https://doi.org/10.2514/1.40490, URL https://doi.org/10.2514/1.40490.

[9] McGrath, C. N., Clark, R. A., and Macdonald, M., "Novel concept of satellite manoeuvre planning using graph theoretical techniques," *Advances in Space Research*, Vol. 67, No. 11, 2021, pp. 3775–3784. https://doi.org/https://doi.org/10.1016/j.asr.2020.06.008, URL https://www.sciencedirect.com/science/article/pii/S0273117720304105, satellite Constellations and Formation Flying.

[10] Globus, A., Crawford, J., Lohn, J., Morris, R., and Clancy, D., "Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach," 2002.

[11] Marinelli, F., Nocella, S., Rossi, F., and Smriglio, S., "A Lagrangian heuristic for satellite range scheduling with resource constraints," *Computers and Operations Research*, Vol. 38, No. 11, 2011, pp. 1572–1583. https://doi.org/https://doi.org/10.1016/j.cor.2011.01.016, URL https://www.sciencedirect.com/science/article/pii/S0305054811000281.

[12] Wang, J., Zhu, X., Yang, L. T., Zhu, J., and Ma, M., "Towards dynamic real-time scheduling for multiple earth observation satellites," *Journal of Computer and System Sciences*, Vol. 81, No. 1, 2015, pp. 110–124.

[13] Lenzen, C., Woerle, M. T., Göttfert, T., Mrowka, F., and Wickler, M., "Onboard planning and scheduling autonomy within the scope of the FireBird mission," *SpaceOps 2014 Conference*, American Institute of Aeronautics and Astronautics, Pasadena, CA, USA, 2014, p. 1759.

[14] Castano, R., Estlin, T., Anderson, R. C., Gaines, D. M., Castano, A., Bornstein, B., Chouinard, C., and Judd, M., "Oasis: Onboard autonomous science investigation system for opportunistic rover science," *Journal of Field Robotics*, Vol. 24, No. 5, 2007, pp. 379–397.

[15] Liu, S., and Yang, J., "A Satellite Task Planning Algorithm Based on a Symmetric Recurrent Neural Network," *Symmetry*, Vol. 11, No. 11, 2019, pp. 1–18.

[16] Wang, C., Li, J., Jing, N., Wang, J., and Chen, H., "A Distributed Cooperative Dynamic Task Planning Algorithm for Multiple Satellites Based on Multi-agent Hybrid Learning," *Chinese Journal of Aeronautics*, Vol. 24, No. 4, 2011, pp. 493–505.

[17] Harris, A., Valade, T., Teil, T., and Schaub, H., "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 611–626. https://doi.org/10.2514/1.A35169, URL https://doi.org/10.2514/1.A35169.

[18] Stephenson, M. A., and Schaub, H., "Optimal Agile Satellite Target Scheduling with Learned Dynamics," *Journal of Spacecraft and Rockets*, Vol. 0, No. 0, 0, pp. 1–12. https://doi.org/10.2514/1.A36097, URL https://doi.org/10.2514/1.A36097.

[19] Herrmann, A., Stephenson, M. A., and Schaub, H., "Single-Agent Reinforcement Learning for Scalable Earth-Observing Satellite Constellation Operations," *Journal of Spacecraft and Rockets*, Vol. 61, No. 1, 2024, pp. 114–132. https://doi.org/10.2514/1.A35736, URL https://doi.org/10.2514/1.A35736.

[20] Wang, H., Yang, Z., Zhou, W., and Li, D., "Online scheduling of image satellites based on neural networks and deep reinforcement learning," *Chinese Journal of Aeronautics*, Vol. 32, No. 4, 2019, pp. 1011–1019.

[21] Nguyen, T. T., Nguyen, N. D., and Nahavandi, S., "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications," *IEEE Transactions on Cybernetics*, Vol. 50, No. 9, 2020, pp. 3826–3839. https://doi.org/10.1109/TCYB.2020.2977374.

[22] Cesa-Bianchi, N., and Lugosi, G., *Prediction, learning, and games*, Cambridge university press, Cambridge, 2006.

[23] Kotłowski, W., "On minimaxity of Follow the Leader strategy in the stochastic setting," *Theoretical Computer Science*, Vol. 742, 2018, pp. 50–65.

[24] Cesari, T., and Colomboni, R., "A nearest neighbor characterization of Lebesgue points in metric measure spaces," *Mathematical Statistics and Learning*, Vol. 3, No. 1, 2021, pp. 71–112.

[25] Wikipedia contributors, "Sentinel-6 Michael Freilich — Wikipedia, The Free Encyclopedia," , 2021. URL https://en.wikipedia.org/w/index.php?title=Sentinel-6_Michael_Freilich&oldid=1057089658, [Online; accessed 26-January-2022].