



Abstract Argumentation for Explainable Satellite Scheduling

1st Cheyenne Powell, 

*Mechanical and Aerospace Engineering
University of Strathclyde
Glasgow, UK
{name.surname}@strath.ac.uk*

2nd Annalisa Riccardi, 

*Mechanical and Aerospace Engineering
University of Strathclyde
Glasgow, UK
{name.surname}@strath.ac.uk*

Abstract—Satellite schedules are derived from satellite mission objectives, which are mostly managed manually from the ground. This increases the need to develop autonomous on-board scheduling capabilities and reduce the requirement for manual management of satellite schedules. Additionally, this allows the unlocking of more capabilities on-board for decision-making, leading to an optimal campaign. However, there remain trust issues in decisions made by Artificial Intelligence (AI) systems, especially in risk-averse environments, such as satellite operations. Thus, an explanation layer is required to assist operators in understanding decisions made, or planned, autonomously on-board. To this aim, a satellite scheduling problem is formulated, utilizing real world data, where the total number of actions are maximised based on the environmental constraints that limit observation and down-link capabilities. The formulated optimisation problem is solved with a Constraint Programming (CP) method. Later, the mathematical derivation for an Abstract Argumentation Framework (AAF) for the test case is provided. This is proposed as the solution to provide an explanation layer to the autonomous decision-making system. The effectiveness of the defined AAF layer is proven on the daily schedule of an Earth Observation (EO) mission, monitoring land surfaces, demonstrating greater capabilities and flexibility, for a human operator to inspect the machine provided solution.

Index Terms—Satellite, Scheduling, Earth Observation, Constraint Programming, Abstract Argumentation, Optimization, Feasibility, Explainability, Explainable Artificial Intelligence, Decision-making, Framework

I. INTRODUCTION

The collation of visual images and data from instruments on-board a satellite are essential for different areas of research for EO missions, such as: vegetation monitoring [1], weather tracking and monitoring [2]–[4], and deep space EO missions [5]. To maximise utilization through the most efficient methods for data throughput of these instruments, it is essential that highly feasible, towards optimal, schedules are created for these satellites [6]–[8].

Wang et al. [9] discussed the current methods for scheduling satellites, which are all based on search algorithms, identifying the following types of methods; exact methods, which are intended for small scale campaigns, but are effective at achieving near optimal solutions; heuristic, that are easily

and quickly implemented, but must be specifically designed for each campaign and cannot guarantee solution quality; metaheuristic, which are highly effective at searching and can be flexibly applied, however have variable performance and require modifying for specific problems; and machine learning, that require large data for training and have limited outputs, yet require minimal maintenance once established.

All methods of solving a satellite schedule require the identification and inclusion of constraints, that determine the parameters by which a satellite must operate. For example, some constraints will restrict when certain activities can be executed, or how often they can be executed, in support of maximising delivery on campaign goals [9]. The mission objectives for each satellite will determine the tasks to be executed, the resources required, the respective orbit and the opportunities for data retrieval by ground stations in fixed locations [7], [10], [11]. Schedules are derived by Ground Station Operators (GSO) and are uploaded to the orbiting platform; however, manually generated mission schedules have their limitations, as they can be labour intensive. As a result, different scheduling techniques are being investigated to assist with scheduling, to reduce human intervention and enhance the robustness in responding to unexpected events [4], [10], [11].

In addition to these unexpected events occurring, when communication is limited or completely unavailable, there is a requirement for scheduling to be performed autonomously on-board. However, there is hesitance in trusting automated systems, due to inconsistent and inaccurate performance across applications such as autonomous driving and robotics [12]. To help explain any decisions made and executed by an autonomous system, Explainable Artificial Intelligence (XAI) can be applied as a means of explaining a system's behaviour. XAI techniques can improve the communication with users by utilising graphs, images, and text to expose the reasoning behind what was executed, what the current state is, and what could happen next [13]. When unexpected situations develop, and where a schedule is updated, XAI is used to provide transparency [14] within a system to build trust and understanding for the generated results retrieved by the GSO. This is achieved by allowing investigation of queries, concerns, and/or impact to the mission objectives [15], [16].

This study was funded by ESA under the OSIP Co-Sponsored PhD activity: “Robust and Explainable Mission Planning and Scheduling (REMPS)” No. 4000132894/20/NL/MH/hm.

Some queries for an EO satellite mission are:

- Is it better to interrupt sequences of repeated actions, or maintain the sequence? For example, should processing occur in the middle of an image taking sequence or after?
- Is the time critical for task 'x' or is task 'x' critical for a certain time?
- Does the on-board memory impact the criticality of a task?
- Why execute an action over another, for example take images vs down-linking vs processing on-board?
- Can the memory limit be relaxed for critical actions?
- If the conditions have been changed, how different will the revised schedule be?

There is no set approach when delivering explanations, however, a use case will determine how much detail is to be provided; for example, researchers will require full and precise explanations as opposed to end users or stakeholders of an organisation, who may require less detailed or different explanations [15], [17], [18].

With an increase in the complexities of computational methods in AI applications, an Argumentation Framework (AF) is a concept used to support explanations comprised of mathematical models [19] that have been defined around the objectives, containing elements of the problem [20]. An AF is represented in different forms of logic, one of which is a graph displaying possible decisions with pre-defined attack properties due to conflicts around the contained elements, resulting in a decision leaning towards the most suitable element for the scenario of the mission [19]–[21].

The detected conflict relations, known as Abstract Argumentation (AA), can influence the conditions to either be improved or changed depending on the properties of elements within a system. This can therefore provide support in explaining how the conditions are affected [22], and thus an AAF has been stated to be created as pairs, containing both arguments and binary relations, where a relation is known as an attack [19]–[21], [23]–[25].

Argument semantics can assist with the labelling and understanding of the logic of attacks and decisions made by a system, through stating whether arguments are accepted, rejected, or undecided [26]. However, this is not considered for this paper as the approach is more advanced argumentation and not the purpose of this paper. The arguments within an AF can have the conditions of their acceptability grouped together as sets, the most commonly used groups being cores and remainders. The set of cores for an argument are the arguments required to be accepted to satisfy the original argument. While conversely, the set of remainders for an argument, are the arguments required to reject the original argument. This is helpful in understanding long chains of dependent arguments but is not appropriate for the scale and nature of AF utilised in this paper [27]. There may be uses for one or both of these methods in future investigations.

In addition to the AAF, there are several other types of AF, some of which are: Bipolar Argumentation Framework

(BAF) - where attack and support relations occur [19], [25], [28] with two extensions: Quantitative Bipolar Argumentation Framework (QBAF) and Probabilistic Bipolar Argumentation Framework (PBAF) [19]; Structured Argumentation Framework (SAF) - where constraints are introduced in the form of preferences between arguments in accordance with the order of these arguments [19], [29]; and Tripolar Argumentation Framework (TAF) - where attack, support, and neutral relations occur to enable interactive recommendations [28]; and Abstract Dialectical Framework (ADF) that specifies the exact conditions by which an argument is accepted (or rejected) through the linking of dependencies, depicted as a directed graph [30]. The principles of ADF influence the methodology of this paper.

To date, and to the best of the authors knowledge, there are no applications of AA to real life engineering problems. Therefore, this paper focuses on applying AA to a previously generated simple EO satellite schedule [16], building on research in other industries, to provide transparent explanations to build trust and enhance the understanding of the generated schedule retrieved by the GSO. The schedule consists of three actions, performed at fixed time intervals, based on the mission objectives and constraints to determine when the actions can be executed.

The contributions of this paper are as follows:

- The formalisation of an EO satellite scheduling problem, including layered constraints to represent real world requirements.
- A novel design and the first application of AA to satellite scheduling.
- Provide results, utilising real world data, demonstrating the success of AA on the scheduling problem, proving it's viability for further investigation.
- Introduce capabilities for XAI on the results, addressing the questions posed in this Introduction, confirming the potential AA has to facilitate XAI interfacing.

The remainder of this paper is divided into five sections, excluding this Introduction:

- Section II contains two subsections:
 - Section II-A provides a summary of the satellite scheduling problem generated by Powell et al. [16].
 - Section II-B defines exchange properties influenced by Cyras et al. [31] and their applications to the satellite schedule derived.
- Section III describes how an AF derived from Cyras et al. [32] can be applied to satellite scheduling.
- Section IV provides graphical representations and explanations of arguments between actions within a satellite schedule for a selected time frame.
- Section V summarizes and concludes the findings of this paper.
- Section VI discusses the future works.

The source code and domain-specific models are available at github.com/strath-ace/smart-xai.

II. BACKGROUND

A. Satellite Scheduling Problem

To generate a schedule for an EO satellite, it is essential to know the actions planned and the timelines, that are driven by constraints, the environmental conditions on Earth and the availability of resources to achieve mission objectives [33]–[35]. Despite the differences between satellite missions, there is a common need for a schedule to be created on-board to compensate for anomalies in the absence of human control, as previously stated in Section I. One approach to creating a schedule is the utilization of a numerical solver.

A problem was formulated by Powell et al. [16], and solved with the use of Google OR-Tools implementing the CP-SAT solver [36] for an EO satellite in sun synchronous orbit, using an existing real world data-set covering a period of 6 months. Three actions were used in the problem formulation: taking of images, processing of images and down-linking of images. Each of these actions were given constraints and a fixed time interval of 5 seconds per occurrence across a given time horizon (24 hrs), which is referred to as an instance of an action throughout this paper. Each instance of an action, based on the hardware capabilities, was assigned a memory value which is monitored while the schedule is being generated by the solver to prevent system memory saturation.

The constraints for the actions are listed below:

- Taking of images:
 - Only occurs when the satellite is over land and in sunlight exposure
 - Will utilize the on-board memory (2.688 GB) for each instance.
 - The number of images in memory will only be reduced when they are down-linked to a ground station.
- Processing of images:
 - Can only process images that have been taken but may occur at any time.
 - Will utilize the on-board memory, with an assumed processing rate of 50 MB/s for 5 seconds per instance, and will not remove any of the memory from the total number of images taken.
 - The number of processed images in memory will only be reduced when down-linking is executed.
 - For every instance of this action, a fraction of an image is processed.
- Down-linking of images:
 - Only occurs when there is direct communication with the assigned ground stations (line of sight) and when there are processed images available in on-board memory.
 - This action, with a data rate of 280 MB/s for 5 seconds per instance, will down-link an equivalent amount of processed data, while deleting the same amount from the images taken, thus removing twice as much from the on-board memory once retrieved by the ground station.

A binary decision matrix X was created (1) where T represents the scheduling time horizon for any time instance i , when any of the possible actions $a_p, a_r, a_d, a_e \in A$ can be executed:

$$X \in \{0, 1\}^{T \times A} \quad (1)$$

Where A represents all actions, containing; a_p for taking an image, a_r for processing an image, a_d for down-linking an image, and a_e generated as idle time when no other actions occur.

The constraint equations excluding the memory constraint are as follows:

$$\sum_{a \in A} X_{i,a} \leq 1 \quad \forall i \in 1, \dots, T \quad (2)$$

$$p_i = \sum_{j=1}^i X_{j,a_p} - \sum_{j=1}^i X_{j,a_d} \frac{D_m}{I_m} \quad \forall i \in 1, \dots, T \quad (3)$$

$$r_i = \sum_{j=1}^i X_{j,a_r} \frac{R_m}{I_m} - \sum_{j=1}^i X_{j,a_d} \frac{D_m}{I_m} \quad (4a)$$

$$r_i \leq p_i \quad \forall i = 1, \dots, T \quad (4b)$$

$$d_i = \sum_{j=1}^i X_{j,a_d} \frac{D_m}{I_m} \quad (5a)$$

$$d_i \leq r_i \quad \forall i = 1, \dots, T \quad (5b)$$

Equation (2) enforces that only one action can be taken at instance i , and to keep track of the images taken and processed in memory, four main equations were created along with their respective constraints. p_i , r_i and d_i represent the total number of remaining images taken in memory (3), the total number of remaining processed images in memory (4a) and (4b), and the total number of images down-linked (5a) and (5b) up to that specific time instance i respectively. Meanwhile the constant values for memory demand of each action, per instance, are represented as I_m (2.688 GB), R_m (250 MB) and D_m (1400 MB), namely: image taking (a_p), image processing (a_r) and image down-linking (a_d) respectively.

It can be noted that (5a) is used in (4a) and (3) to remove the total number of images down-linked up to that point in time. Since the constant value for memory demand D_m for every down-linked action instance a_d equates to a fraction of the memory of an image I_m ; the total occurrence throughout is subtracted from the total images taken and processed resulting in the number of images and processed left in memory (p_i and r_i). Furthermore, only images that have been processed can be down-linked (5b) as defined by the constraints.

As can also be seen in (4a), since the memory demand for every instance of action a_r , known as R_m , is a fraction of the original image memory I_m ; subtracting the total down-linked images up to that instance in time is done to provide the total number of remaining processed images in memory. Also, in (4b), there must be at least 1 unprocessed image in memory for this action to occur as defined by the constraints.

Additionally, to calculate the total memory at any instance m_i requires the number of remaining taken images in memory (p_i) and the remaining processed images in memory (r_i) followed by converting the number of images remaining in the form of memory (6a) followed by an additional constraint (6b) ensuring the memory does not exceed the maximum memory available on-board M_{max} . The memory constraint is therefore:

$$m_i = I_m(p_i + r_i) \quad \forall i = 1, \dots, T \quad (6a)$$

$$m_i \leq M_{max} \quad \forall i = 1, \dots, T \quad (6b)$$

Equations (3) - (5) were used to create objective function (7) for the schedule, aiming to maximise the weighted sum of the three actions, excluding a_e . The scheduling problem can be formulated as:

$$\max \left(\sum_{i=1}^T X_{i,a_p} + \sum_{i=1}^T X_{i,a_r} + \sum_{i=1}^T 2X_{i,a_d} \right) \quad (7)$$

To better understand how a schedule is generated with the use of the objective equation, environmental conditions, and constraints that can affect each action execution, two figures were created:

- Fig. 1 represents all the possible execution times for each action only considering the environmental conditions, before the constraint equations and objective function are applied:
 - a_p (shaded red), with respect to the light exposure (shaded yellow) and land visibility (shaded green).
 - a_r (shaded blue), may happen at any time, as previously stated.
 - a_d (shaded orange), may occur when there is line of sight with the assigned ground stations (shaded grey).
 - a_e , should happen when no other action occurs but isn't shown as a_r can be seen to be in execution across the time horizon.
- Fig. 2 represents an example of a generated schedule on the first day where the constraints (including the memory constraints), and the objective equation ((2) - (7)), are applied to the actions from Fig. 1. The broken lines highlighted represent the start and end times of each action, and show a_p starting initially, followed by a_r (4b), then a_d being executed, and a_e (shaded purple) occurring filling in the available slot.

It can be noted, the environmental conditions are suited for a_d to be executed sooner, but could not have occurred as it would have violated the constraint based on (5b). Additionally, at each instance, the maximum memory (M_{max}) would not have been exceeded at any time.

Following the generation of the schedule, exchange properties were investigated to see how they can be applied to a satellite schedule.

B. Application of exchange properties for satellite scheduling

In generating an AAF for the previously defined scheduling problem, the approach taken was to investigate the applicability of the Single Exchange Property (SEP) and Pairwise

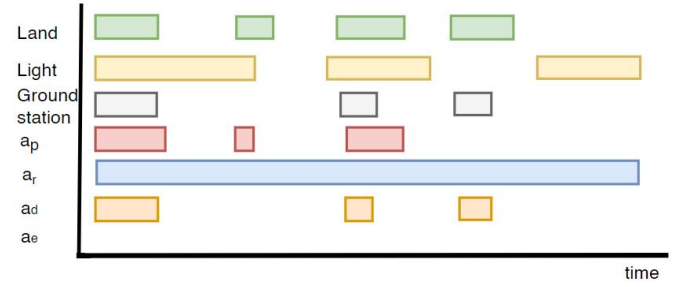


Fig. 1. A sample schedule for all the possible action executions based on the environmental conditions.

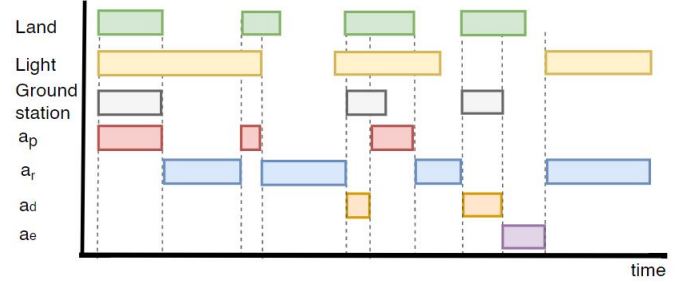


Fig. 2. A sample schedule generated based on the environmental conditions and constraints applied to each action.

Exchange Property (PEP) for the targeted application. These definitions have been adapted from the work of Cyras et al. [31], where their objective was to determine the minimum makespan of a schedule by minimizing the last machine completion time, followed by examples on how it can be applied to a nurse roster.

The authors defined SEP as a single exchange of any critical job with another job between machines throughout a schedule to improve the given schedule. While PEP entails an exchange of multiple critical jobs with other jobs between machines to gain improvements [31].

These definitions have been adapted to apply to the satellite scheduling problem introduced in Section II-A.

Initially, an equation needs to be defined that can be applied to both concepts derived by Cyras et al. [31] and Powell et al. [16]. We are noting that:

$$v^* = \min_{a \in A} v_a \quad (8)$$

where v_a is the memory required by action a during any time, hence v^* is the smallest memory value required by the considered actions, excluding a_e , as it has no value; unless it undergoes an attack from another action or is undergoing PEP, which is later explained in Section III, with results shown in Section IV. Equation (8) is used as a representation of any of the three memory variables I_m , R_m and D_m , as they have a fixed memory value at any time instance, as previously defined in Section II-A.

Definition II.1. An action $a \in A$ is said to be critical at time

interval $i \in \{1, \dots, T\}$ if:

$$X_{i,a} = 1 \wedge M_{max} - v^* \leq m_i \leq M_{max}$$

Meaning that an action is defined as critical during a specified instance, where if any other action were to replace that action and this change is cascaded throughout the schedule, it will not satisfy the on-board maximum memory constraint.

Definition II.2. SEP is satisfied by a schedule S iff for every critical action $a \in A$ at instance i , it holds that, for any $a' \neq a$

$$m_i - m'_i > v_{a'} - v_a \quad \wedge \quad m'_j \leq M_{max}, \forall j > i$$

where m'_i and m'_j is the value of memory at time interval i and j respectively, if action a' is chosen instead of action a at time interval i .

Once an action v_a has been replaced by any of the other three actions $v_{a'}$, an alternate memory m'_i is created at that instance and therefore an alternate memory m'_j following this action is generated throughout the schedule, and while within M_{max} will satisfy SEP.

Definition II.3. PEP is satisfied by a schedule S iff for every critical action $a \in A$ at instance i , it holds that, for any $a' \neq a$, and any $j \neq i$:

$$m_i - m'_j > v_{a'} - v_a \quad \wedge \quad m'_k < M_{max}, \forall k$$

Therefore an action v_a is exchanged with any of the other three actions $v_{a'}$ at a different instance j . Two alternate memory values are created (m'_k and m'_j) where m'_j is generated following the exchange and m'_k is the memory value being updated throughout each of the following actions to ensure M_{max} is satisfied, thus satisfying PEP.

Overall SEP ensures that the feasibility and criticality of action a , with respect to the memory constraint, is not worsened by the substitution at time interval i and in all the following time intervals. Similarly PEP ensures that the swapping of a critical action with another one does not invalidate the memory constraint across the whole time horizon.

The authors Powell et al. [16] generated an optimal schedule S^* utilising the CP-SAT solver, from an initial first guess S , modelled with heuristics from problem constraints. Similar to Cyras et al. [31], we can define a negative decision set as the subset of decision space $T \times A$ where action constraints, namely equations: (3), (4a), (4b), (5a), and (5b) hold. These are the constraints that were derived from the position of the satellite in time and the possibility of performing a particular action.

Definition II.4. The negative fixed decision set is defined as:

$$D^- = T^- \times A^- \subseteq T \times A$$

such that a feasible schedule S satisfies:

$$(i, a) \in D^- \rightarrow X_{i,a} = 0$$

The definition of the negative decisions can also capture extreme situations, such as memory being saturated during time i and the next opportunity for down-link is at a time

interval $j > i$. In this case, the negative decision set is transformed into:

$$D^- = T^- \times A^- \cup \{(k, a) : i < k < j\}.$$

Similarly, the positive fixed decisions can also be defined for this specific use case. These may be actions enforced by the GSO in support of EO campaign goals.

Definition II.5. The positive fixed decision set is defined as:

$$D^+ = T^+ \times A^+ \subseteq T \times A$$

such that a feasible schedule S satisfies:

$$(i, a) \in D^+ \rightarrow X_{i,a} = 1$$

Here the decisions will be kept, resulting in a schedule being derived around these decisions, generating a feasible schedule.

III. ABSTRACT ARGUMENTATION (AA)

AA is a mathematical framework that analyses the conflicts between two or more arguments [20], [21], [25]. Arguments may represent a particular realisation of decisions within a schedule. If changes were made dynamically within a schedule due to AA occurring, this can influence the execution time of the activities within the schedule and thus create further arguments and attacks and generate a structured argument [37]. In combining the concepts of SEP and PEP with the concept of AA, when an exchange occurs between two or more actions, this is considered an attack.

Therefore, the constraints for attacks created for this paper are:

- Any action can attack another except themselves.
- When in SEP, a_e must not attack another action, however can be attacked by another.
- When in PEP, a_e can attack another action.
- For PEP, only exchanges between any two actions within the schedule, excluding themselves, may occur across the schedule.

Fig. 3 is a representation of what happens when there is an attack within a schedule S at instance i , with an existing memory (m_{i1}). Individual memory values V_{ap} , V_{ar} , and V_{ad} , also referred to as v'_i , have arrows coloured pink, blue, and green respectively representing every action attack, a_p , a_r , and a_d . The constraints equation (6) is used to ensure there is no breach in M_{max} at m_{i1} , therefore complying with SEP, and an alternate memory m'_i is generated, creating alternate memories for proceeding times m'_j for $(m_{i2}, m_{i3}, m_{i4} \dots m_j)$, further generating $(m'_{i2}, m'_{i3}, m'_{i4} \dots m'_j)$ based on the following actions scheduled, as stated in Definition II.2.

In this example, it is important to note: V_{ap} , V_{ar} , and V_{ad} will only occur when the action at m_i is a_e , where the three other actions will attempt to replace this existing action. Otherwise for SEP, there will only be two actions attacking; e.g. if a_p is presently scheduled to take place at instance i then actions a_r and a_d will attack.

The concept of PEP in Fig. 4 represents an overview of how pairwise exchanges may occur within a schedule

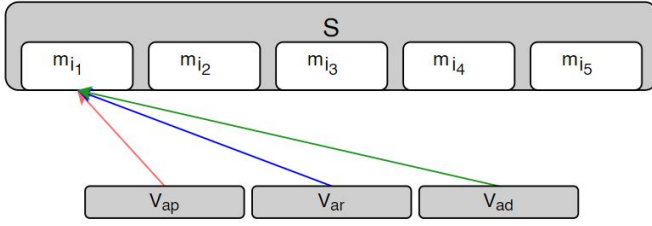


Fig. 3. An overview of the argumentation process across one time interval representing the occurrence of a single exchange of an action with memory variables V_{ap} , V_{ar} , and V_{ad} within a schedule S to alter the memory m_{i1} at that instance.

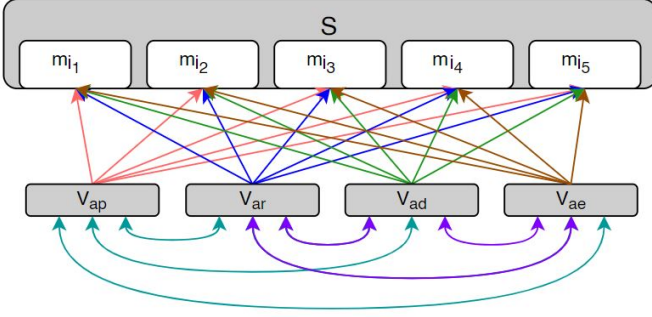


Fig. 4. An overview of the argumentation process across each time interval representing what occurs during a pairwise exchange between actions within a Schedule S , impacting the existing memory m_{i1} , m_{i2} , m_{i3} , m_{i4} , and m_{i5} , on-board a satellite by the respective memory values V_{ap} , V_{ar} , V_{ad} and V_{ae} of an action following an attack.

S between any two non-identical actions scheduled to be executed at different times. The existing memory values ($m_{i1}, m_{i2}, m_{i3} \dots m_j$) prior to an exchange of actions remain the same. Following an exchange across the schedule, alternate memories m'_j and m'_k , as defined in Definition II.3, are created only on the condition that M_{max} is not breached, as stated in (6b). It can be noted, in the figure there are four V_a variables representing each of the four actions a_p , a_r , a_d , and a_e being interchanged with each other by the teal and purple bi-directional arrows. The action at i will determine which a will be exchanged and then applied to create an m'_j and these arrows are represented with colours pink, blue, green, and brown for values V_{ap} , V_{ar} , V_{ad} and V_{ae} respectively. Therefore, meaning an attack has been created that led to a decision, resulting in a decision vector $X_{i,a}$, the memory profile $m_j \forall j \geq i$, and the final objective function value.

A binary attack (\rightsquigarrow) is defined as an action that can occur within the schedule at a specified instance over an existing action currently scheduled for the same time. However, this is based on the memory availability and the position of the satellite, which will determine whether or not the action is feasible. Represented in Fig. 5, if action a_p , for example, was scheduled at i , it would be analysed for viability of attacks by actions a_r , a_d and a_e ; while a_r would be analysed by a_p , a_d and a_e ; a_d by a_p , a_r and a_e ; and a_e by a_p , a_r and a_d respectively, with all actions represented by circles shaded grey.

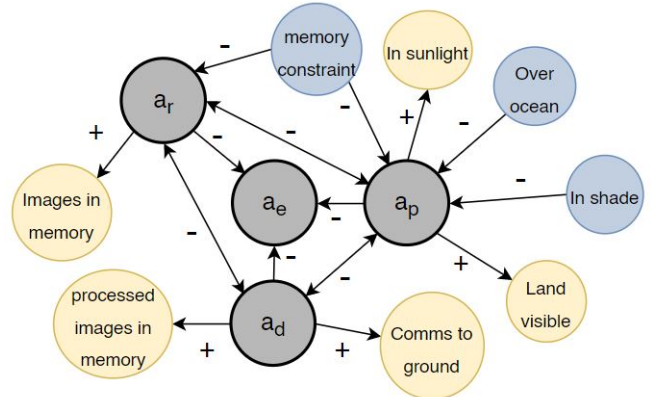


Fig. 5. Conditions involved with attacks a_p (Image taking), a_r (Image Processing), a_d (Down-linking), and a_e (Idle time) during time i and the conditions affecting '-' and supporting '+' these actions from occurring.

Both negative and positive arguments are represented in Fig. 5, with an arrow pointing away from the blue shaded circles, with negative '-' conditions preventing the actions from occurring. The arrows pointing towards the yellow shaded circles are the supportive '+' conditions for an action to occur based on the conditions; with the exception of idle time, a_e , as that action has no memory value and is only generated when no other actions may occur, as stated in Section II-A. This conflict and support concept was derived from the definition of BAF, as described in Section I. The environmental conditions, such as land visibility and ground station access (highlighted as constraints in Section II-A), control the possibilities of the execution of the actions; as a result Fig. 5 is represented to best visualise how each action execution is impacted by the conditions.

In applying the constraints along with SEP and PEP, it can be noted, $a_{i,j}$ with the argument "An assignment of an action j to time instant i ", by applying the principles of AA to feasibility, the following definition is:

Definition III.1. The feasibility AF, $(Args_F, \rightsquigarrow_F)$ is defined as:

- $Args_F = \{a_{i,j} : (i,j) \in (T \times A) \setminus D^-\}$
- $a_{i,j} \rightsquigarrow_F a_{k,l}$ iff $j \neq l, i = k$
- $a_{i,j} \rightsquigarrow_F a_{k,l}$ iff $m_q > M_{max}$ for $q > i$ if $i < j$, or for $q > j$ if $j < i$.

This therefore means, in addition to the assignment of the action within the decision matrix (first point), the second point represents an action attacking another for the available time slot, as only one action can occur at any time instance. The third point represents an action attacking another only when the memory is breached, if the duration of i is not long enough for the action to be completed while the action has occurred before time q ; or if the action ends before the given duration of i , while it has occurred before time q . This means another action would be preferred to prevent the memory breach and utilize the available time efficiently.

IV. RESULTS

An EO satellite schedule in sun synchronous orbit was generated with the use of the CP-SAT solver derived by Powell et al. [16]. The generated schedule spanned over a time frame of 14 days, from the 6 months data-set. A day from the 14 days was selected excluding the first, as the schedule was initialized with an empty memory, on this day.

A. SEP Results

Starting with SEP, each action scheduled was attacked by any other action, apart from itself and action a_e , and a violation of SEP occurred when no other action could be executed for the specified time instance i . Whenever an attack occurred the replacement of the action at i affected the memory profile from that point onward, thus, depending on the alternate memory created, this may result in a solution being either feasible or infeasible, with respect to the maximum memory constraint at instance j' , as shown in Definition II.2.

To visualize the attacks of all actions, a 2 hour period (equating to 1440, 5 second interval data points) was selected out of the day and shown in Fig. 6, to produce a gantt chart when each action was scheduled to be executed. Below the scheduled actions (a_p , a_r , a_d , and a_e) are the infeasible results for every attack of each action made at each instance, as a representation of M_{max} being breached. As previously mentioned in Section II-B, for every instance, the effect of each attack was cascaded through the onward schedule to assess the infeasibility of the attack. It can be noted no infeasible results were recorded for a_d as there were enough processed images in the memory during this time period cross referenced to shaded region in Fig. 7.

Fig. 7, shows the overall memory profile, with all three main actions displaying the number of images held in memory at any instance in time throughout the day on the left, and the total memory utilized labelled on the right. At 21 : 04 hrs, action a_r was attacked by a_p , as highlighted in a blue shaded region, followed by a broken vertical line at m'_j where the memory breached M_{max} , represented by a horizontal green line in the figure. It can be seen earlier that day the memory approached saturation, however in that instance down-linking, coloured in red, occurred immediately after; reducing the processed and original images stored in memory on-board the satellite, coloured blue and purple respectively, thus reducing the utilized memory. The count for the number of images down-linked resets after every 3000 instances to improve the readability of the graph plotted as this number will only increase. Due to the time horizon shown in this figure it is challenging to see the variance in the memory following the attack. As a result, Fig. 8 is provided to show a magnified view of what happened at i and the effect it had on the memory at j' . This means, according to the definition, SEP was not satisfied and an alternate action would be tested by the system to determine if there is another schedule that can be derived.

In addition to the figures provided, a calculation was done to retrieve a total percentage of all the infeasible solutions over the period of the same day (Fig. 7) as shown in Table

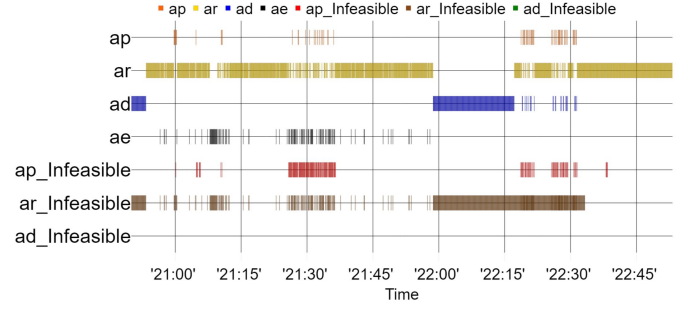


Fig. 6. Summary of infeasible solutions when an action attack occurs.

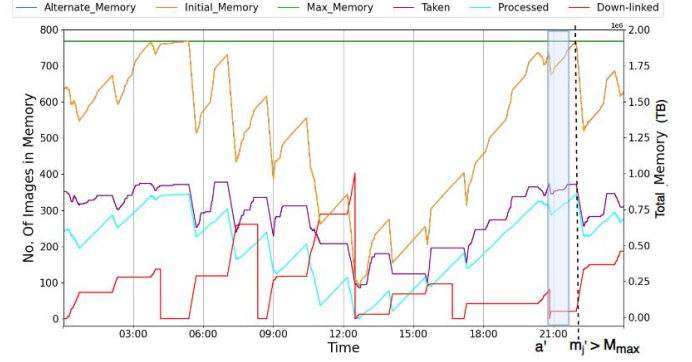


Fig. 7. Results of a_p attack on a_r resulting in an alternate memory m'_i at time instance i followed by the updated memory m'_j exceeding the available memory on-board M_{max} .

I. This means the scheduled actions (a_p , a_r , a_d) including a_e , were individually attacked by each of the other actions excluding themselves. Thus, across 17,278 data points, when a_r , a_d and a_e were attacked by a_p , this generated 74% (2897 possibilities), 16% (630 possibilities), and another 6% (244 possibilities) respectively out of a total of 3921 exchange possibilities, with the remaining 4% returning feasible solutions.

Meanwhile a_r attacking a_p generated 0% due to every exchange being feasible as this action exchange reduces the memory usage, a_d generated 44% (1923 possibilities), and a_e generated 19% (821 possibilities) out of a total of 4324 exchange possibilities, with the remaining 37% returning feasible solutions.

Finally, when a_d attacked a_p , a_r , and a_e respectively, the exchanges generated no infeasible possibilities, meaning it could have been exchanged at any of the 2082 instances, reducing the memory by the value of the action V_{ap} or V_{ar} as it has been replaced, and V_{ad} as a_d represents the removal of memory by V_{ad} .

B. PEP Results

Following the application of SEP, PEP was then investigated to see the effects it would have on the generated satellite schedule; using the same day and data to see what the implications would be and to determine whether it is a viable approach for applying it to AA for this type of problem. Fig. 9 was created in the form of an $n \times m$ matrix where the actions

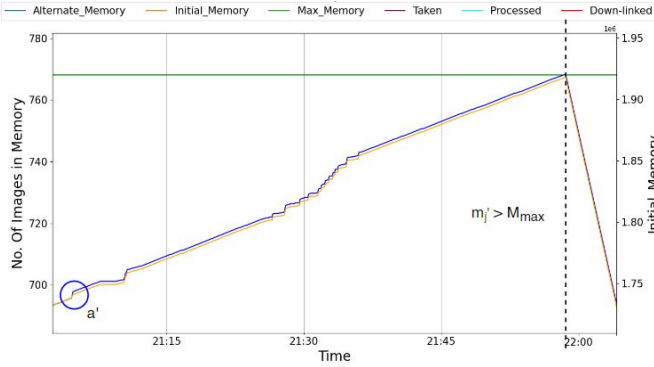


Fig. 8. A magnified view of Fig. 7 showing where memory at time instance j' breached the maximum on-board memory M_{max} following an attack at instance i .

TABLE I
PERCENTAGES OF ACTION EXCHANGE INFEASIBILITIES OVER A PERIOD
OF 1 DAY

Action Exchanges	Scheduled Actions			a_e
	a_p	a_r	a_d	
	%	%	%	
a_p	-	74	16	6
a_r	0	-	44	19
a_d	0	0	-	0

are scheduled for time i (represented as 5 second intervals) throughout a day. The time range displayed in the image starts at 02 : 12 : 31 hrs ending at 02 : 13 : 41 hrs; placed in both the n rows and m columns generating a grid containing cells with numbers represented as actions: (a_p , a_r , a_d and a_e) as (0, 1, 2, -1) respectively. When these cells, containing any two actions excluding the action itself (shaded purple), overlapped at any instance, a pairwise exchange was initiated resulting in the actions attacking each other at the other actions time slot. Upon completion of the swap, as described in Definition II.3 and Section III, a violation check was done, initially at the earlier scheduled action, where a revised m'_j was created. The change in memory, as with SEP, was carried throughout the schedule, while with each iteration the alternate memory was checked until the second action, that was previously exchanged, was approached by the system. Another memory check was done at this point to determine if the second revised action was appropriate. On the condition it was, the scan resumed until either memory has been breached, or the end of the schedule has been reached. When a breach occurred at any point, the scan was terminated and the cell was shaded red, treating that exchange as infeasible; whereas, when PEP was satisfied, the cell was shaded green to confirm the alternate schedule was feasible.

Following the observation of these results, the questions raised in Section I were considered; the answers to which may be based on the hardware capabilities of the system, as well as the amount of flexibility the system is allowed by the GSO for making decisions. This may determine altered or new definitions of the constraints, as demands may change during

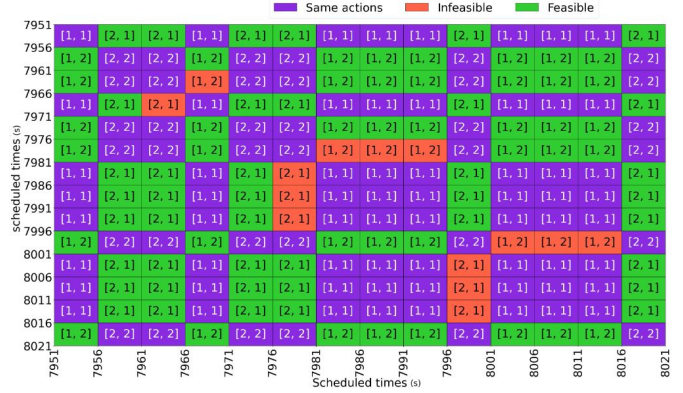


Fig. 9. An example showing the implications of pairwise exchange occurring between any two actions within a 70 seconds window of a satellite schedule.

a satellites' mission. However, the concepts of SEP and PEP were used to see what would happen if action 'x' was replaced with another and how different a schedule may appear based on the actions replaced.

Overall, this AA approach was deemed suitable and useful to assist an explainable layer in providing explanations to the end user, as XAI, noting that AA was used in this paper as a means of explaining the schedule created by the solver and not how the solver converged to the schedule. This may be applied to, but is not limited to the following:

- Provide a user interface with the generated schedule to the end user, allowing them to interact and query the system.
- Provide computational results of the reasoning behind the generated schedule.
- Provide detailed conditions representing conflicts with the environment and conditions on-board a satellite, as shown in Fig. 5.
- Provide an instantaneous reflection of the exchange properties of SEP and/or PEP, as shown in Fig. 3 and Fig. 4, based on the users queries.
- Generate a representation of a decision matrix to enable the user to see the decisions made that were influenced by the conditions.

V. CONCLUSION

AA was applied to an EO satellite scheduling problem to achieve feasibility through the form of relational attacks of scheduled actions by other substituting actions at different time intervals throughout a day. Each action within the generated schedule was subject to constraints that needed to be satisfied throughout attacks, which resulted in an immediate memory comparison to the original schedule when SEP was applied. PEP enabled two actions at any point in time to be exchanged with each other, on the condition that the on-board memory stayed within its limit. When a breach occurred, depending on the exchange property, visual techniques were provided to see the impact the schedule would experience. To summarize this paper, AA has been shown to be useful to assist with the improvements of satellite scheduling, which may be used

within an explainable layer of an autonomous scheduling system, which leads to more advanced applications of XAI.

VI. FUTURE WORKS

There remains further techniques to explore, such as argumentation semantics and sets, and other AF types, namely TAF, that could add additional detail and accuracy to an expanded problem. The problem could progress in exploring the use of AA for more complex campaigns aligned to real world problems, solving for a constellation of satellites for instance. These deeper investigations could then add further scope to develop a functional prototype explainable layer, which would include the exploration of effectiveness in visual displays and prompts, how to provide XAI for different implementations of scheduling solvers, and the use of Natural Language Processing (NLP) for facilitating interactions between the user and the system.

ACKNOWLEDGEMENTS

The authors would like to warmly thank Dimitris Kardaris (ESA) and Simone Fratini (Solenix) for all the insightful discussion about mission operations constraints and opportunity for explainability.

REFERENCES

- [1] M. Gazzea, M. Pacevicius, D. O. Dammann, A. Sapronova, T. M. Lunde, and R. Arghandeh, "Automated power lines vegetation monitoring using high-resolution satellite imagery," *IEEE Transactions on Power Delivery*, vol. 37, no. 1, pp. 308–316, 2022.
- [2] M. W. Maier, F. W. Gallagher, K. St. Germain, R. Anthes, C. Zuffada, R. Menzies, J. Piepmeier, D. Di Pietro, M. M. Coakley, and E. Adams, "Architecting the future of weather satellites," *Bulletin of the American Meteorological Society*, vol. 102, no. 3, pp. E589–E610, 2021.
- [3] W. Schreiner, J. Weiss, R. Anthes, J. Braun, V. Chu, J. Fong, D. Hunt, Y.-H. Kuo, T. Meehan, W. Serafino, J. Sjöberg, S. Sokolovskiy, E. Talaat, T. Wee, and Z. Zeng, "Cosmic-2 radio occultation constellation: First results," *Geophysical Research Letters*, vol. 47, no. 4, p. e2019GL086841, 2020. e2019GL086841 2019GL086841.
- [4] X. Wang, G. Song, R. Leus, and C. Han, "Robust earth observation satellite scheduling with uncertainty of cloud coverage," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 2450–2461, 2020.
- [5] J. H. Jiang, A. J. Zhai, J. Herman, C. Zhai, R. Hu, H. Su, V. Natraj, J. Li, F. Xu, and Y. L. Yung, "Using deep space climate observatory measurements to study the earth as an exoplanet," *The Astronomical Journal*, vol. 156, p. 26, 6 2018.
- [6] K. Sundar, J. Qin, S. Rathinam, L. Ntamo, S. Darbha, and C. Valicka, "Algorithms for a Satellite Constellation scheduling Problem," *IEEE International Conference on Automation Science and Engineering*, vol. 2016-November, pp. 373–378, 2016.
- [7] S. Augenstein, A. Estanislao, E. Guere, and S. Blaes, "Optimal scheduling of a constellation of Earth-imaging satellites, for maximal data throughput and efficient human management," *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, vol. 2016-January, no. Icaps, pp. 345–352, 2016.
- [8] D. H. Cho, J. H. Kim, H. L. Choi, and J. Ahn, "Optimization-based scheduling method for agile earth-observe satellite constellation," *Journal of Aerospace Information Systems*, vol. 15, no. 11, pp. 611–626, 2018.
- [9] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3881–3892, 2020.
- [10] F. Xhafa, J. Sun, A. Barolli, A. Biberaj, and L. Barolli, "Genetic algorithms for satellite scheduling problems," *Mobile Information Systems*, vol. 8, no. 4, pp. 351–377, 2012.
- [11] H. Fan, Z. Yang, S. Wu, X. Zhang, J. Long, and L. Liu, "An efficient satellite resource cooperative scheduling method on spatial information networks," *Mathematics*, vol. 9, no. 24, pp. 1–23, 2021.
- [12] A. Rastogi and K. E. Nygard, "Trust and security in intelligent autonomous systems," *International Journal of Computers and their Applications*, vol. 26, no. 1, 2019.
- [13] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, "Xai𠅎xplainable artificial intelligence," *Science Robotics*, vol. 4, no. 37, p. eaay7120, 2019.
- [14] M. Kuk, S. Bobek, and G. J. Nalepa, "Explainable clustering with multi-dimensional bounding boxes," *2021 IEEE 8th International Conference on Data Science and Advanced Analytics, DSAA 2021*, 2021.
- [15] A. Kotriwala, B. Kloepper, M. Dix, G. Gopalakrishnan, D. Ziobro, and A. Potschka, "XAI for operations in the process industry - Applications, theses, and research directions," *CEUR Workshop Proceedings*, vol. 2846, 2021.
- [16] C. Powell and A. Riccardi, "Towards explainability of on-board satellite scheduling for end user interactions," Oct. 2021. 72nd International Astronautical Congress, IAC 2021 ; Conference date: 25-10-2021 Through 29-10-2021.
- [17] S. Dhanorkar, C. T. Wolf, K. Qian, A. Xu, L. Popa, and Y. Li, "Who needs to know what, when?: Broadening the Explainable AI (XAI) Design Space by Looking at Explanations across the AI Lifecycle," *DIS 2021 - Proceedings of the 2021 ACM Designing Interactive Systems Conference: Nowhere and Everywhere*, pp. 1591–1602, 2021.
- [18] S. C.-H. Yang and P. Shafto, "Explainable Artificial Intelligence via Bayesian Teaching," *Neural Information Processing Systems Workshop: Teaching Machines, Robots, and Humans*, no. Nips, 2017.
- [19] A. Vassiliades, N. Bassiliades, and T. Patkos, "Argumentation and explainable artificial intelligence: A survey," *Knowledge Engineering Review*, vol. 36, no. 2019, 2021.
- [20] T. J. Bench-Capon and P. E. Dunne, "Argumentation in artificial intelligence," *Artificial Intelligence*, vol. 171, no. 10-15, pp. 619–641, 2007.
- [21] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, no. 2, pp. 321–357, 1995.
- [22] M. Ulbricht and J. P. Wallner, "Strong Explanations in Abstract Argumentation," *35th Conference on Artificial Intelligence*, pp. 6496–6504, 2021.
- [23] R. Baumann and G. Brewka, "Expanding argumentation frameworks: Enforcing and monotonicity results," *Frontiers in Artificial Intelligence and Applications*, vol. 216, pp. 75–86, 2010.
- [24] Z. G. Saribatur and J. P. Wallner, "Existential Abstraction on Argumentation Frameworks via Clustering," in *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 549–559, 11 2021.
- [25] P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre, *Handbook of Formal Argumentation*. London, England: College Publications, 2018.
- [26] M. Caminada, "A gentle introduction to argumentation semantics," *Lecture material, Summer*, 2008.
- [27] M. O. Moguillansky, "A study of argument acceptability dynamics through core and remainder sets," pp. 3–23, 2016.
- [28] A. Rago, O. Cocarascu, C. Bechliyanidis, D. Lagnado, and F. Toni, "Argumentative explanations for interactive recommendations," *Artificial Intelligence*, vol. 296, p. 103506, 2021.
- [29] S. Modgil and H. Prakken, "The ASPIC + framework for structured argumentation: A tutorial," *Argument and Computation*, vol. 5, no. 1, pp. 31–62, 2014.
- [30] G. Brewka and S. Woltran, "Abstract dialectical frameworks," pp. 102–111, 2010.
- [31] K. Čyras, D. Letsios, R. Misener, and F. Toni, "Argumentation for explainable scheduling (Full Paper with Proofs)," *arXiv*, 2018.
- [32] K. Čyras, M. Lee, and D. Letsios, "Schedule explainer: An argumentation-supported tool for interactive explanations in makespan scheduling," in *Explainable and Transparent AI and Multi-Agent Systems* (D. Calvaresi, A. Najjar, M. Winikoff, and K. Främling, eds.), (Cham), pp. 243–259, Springer International Publishing, 2021.
- [33] CCSDS, "Mission Planning and Scheduling," Tech. Rep. June, CCSDS - Consultative Committee for Space Data Systems, Washington, DC, USA, 2018.
- [34] F. Xhafa and A. W. Ip, "Optimisation problems and resolution methods in satellite scheduling and space-craft operation: a survey," *Enterprise Information Systems*, vol. 15, no. 8, pp. 1022–1045, 2021.
- [35] S. A. Chien, M. Johnston, J. Frank, M. Giuliano, A. Kavelaars, C. Lenzen, and N. Policella, "A generalized timeline representation, services, and interface for Automating Space Mission Operations," *SpaceOps 2012 Conference*, pp. 1–17, 2012.

- [36] L. Perron and V. Furnon, “Or-tools,” *Google.[Online]*. Available: <https://developers.google.com/optimization>, 2019.
- [37] A. Borg and F. Bex, “Enforcing Sets of Formulas in Structured Argumentation,” in *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 130–140, 11 2021.