Appears in Journal of Aerospace Information Systems (JAIS), 2024. https://doi.org/10.2514/1.I011323

A Utility-Driven Approach to Onboard Scheduling and Execution for an Autonomous Europa Lander Mission

Joseph A. Russino [*], Daniel Wang[†], Caleb Wagner[‡], Gregg Rabideau[§], Faiz Mirza[¶], Connor Basich[‖], Cecilia Mauceri[**], Philip Twu[††], Glenn Reeves[‡‡], Grace Tan-Wang[§§], Steve Chien[¶¶]
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, California 91109

Jupiter's icy moon Europa is among the most promising locations to explore for signs of extraterrestrial life in our solar system. However, searching for biosignatures on the surface of Europa presents an unprecedented set of challenges: immense uncertainty, limited energy, and few opportunities for operator feedback. Effectively carrying out a robotic science campaign under these conditions will require a system with a greater degree of autonomy than any planetary exploration mission to date. In particular, on-board scheduling and execution is required for robustness to: uncertainty in surface conditions, variation in lander performance, and science discoveries to maximize the quantity and quality of science data returned to Earth during a fixed, limited lander lifetime. Here we represent a proposed Europa Lander surface mission as a utility-driven Hierarchical Task Network and establish through analysis that onboard autonomy using automated mission planning using execution feedback and predictive task models results in mission execution that is more consistently productive compared to traditional static approaches. We design a simulated on-board autonomy framework built by integrating two software components – Multi-Mission Executive (MEXEC) and Europa Lander Autonomy Prototype (ELAP) – to properly simulate the Europa Lander domain, and demonstrate empirically that the proposed planning and execution system is capable of commanding a set of realistic surface scenarios as part of a larger Europa Lander surface autonomy software prototype. We expect that an approach to scheduling and execution that is grounded in decision theory will be an enabling technology for future tightly-constrained planetary surface missions.

[*]Corresponding Author; MEXEC Team Lead, Europa Lander Pre Project; Senior Member of Technical Staff, Artificial Intelligence Group; Correspondence email: joseph.a.russino@jpl.nasa.gov

[†]Member of Technical Staff, Artificial Intelligence Group

[‡]Prototype Deputy Team Lead, Autonomy Prototype, Europa Lander Pre project; Member of Technical Staff, Artificial Intelligence Group

[§]Principal Member of Technical Staff, Artificial Intelligence Group

[¶]Senior Member of Technical Staff, Artificial Intelligence Group

[‖]Member of Technical Staff, Artificial Intelligence Group

[**]Prototype Development Lead, Autonomy Prototype, Europa Lander Pre project

[††]Autonomy Architect, Autonomy Prototype, Europa Lander Pre project; Task Manager

[‡‡]System Level Autonomy Lead, Autonomy Prototype, Europa Lander Pre project; JPL Fellow

[§§]Conops, Mission System, and Ground System Lead, Autonomy Prototype, Europa Lander Pre project

[¶¶]Element Lead for Scheduling and Execution, Autonomy Prototype, Europa Lander Pre project; JPL Fellow, Senior Research Scientist

I. Introduction

AI planning for robotic applications must often address variation in execution and uncertainty in the accuracy of environment models. In space-based applications, this can be especially challenging when the environment is largely unknown, reducing the quality of our *a priori* models of the world. To address these problems, we propose an integrated approach to planning and execution in an unknown, unpredictable environment that is grounded in decision-theoretic reasoning, i.e. driven by *expected return*, but still adheres to severe onboard computing constraints.

The primary motivation for this work is a mission concept to perform *in situ* analysis of samples collected from the surface of the Jovian moon Europa [1]. The ultimate goal for a Europa Lander would be to analyze surface material and communicate the resulting data products back to Earth. The mission variability results from decisions of: (1) where to excavate and sample and how many samples to collect in the face of unpredictable rates of progress, as well as (2) which data to downlink and how much to summarize in light of whether biosignatures are detected, and if so, how strong the indication of biosignatures is. To reward accomplishment of these goals, we assign utility to tasks such as sample excavation and seismographic data collection, but the overwhelming majority of the mission utility is not awarded until the lander communicates the data down to Earth.

Unlike prior NASA missions, *a priori* domain knowledge is severely limited and uncertain, and communication with Earth is limited by long blackout periods (over 42 hours out of every 84 hours). Consequently, a successful mission requires a planning and execution framework that can operate autonomously for extended periods of time, is robust to unprecedented levels of uncertainty, and is still capable of maximizing its overall utility. Additionally, because of the harsh radiation environment at Europa, mission lifetime and onboard computing are severely limited.[1] [2]

On the other hand, the Europa Lander mission concept has a well defined set of actions the lander must perform in order to execute the mission. Our planning algorithm leverages this domain-specific mission knowledge by making use of hierarchical task networks (HTNs) which have been successfully used in many real-world applications (a complete description can be found in Section II.C), and using heuristic-guided search to examine various task combinations to maximize mission utility. We further improve the robustness of our approach with the inclusion of two on-board autonomy capabilities for handling uncertainty during execution: *flexible execution* and *replanning with plan optimization*.

To validate our approach, we compare four approaches to planning on the Europa Lander problem including approaches used in prior missions – a static plan without failure recovery mechanisms, a static plan with ground input for failure recovery (similar to current Mars Rover operations) [3], flexible execution without replanning, and flexible execution with replanning optimization – and examine the effect that onboard autonomy (flexible execution and replanning with plan optimization) has on the utility of these approaches. We empirically demonstrate that each

[1]As a point of reference, the RAD750 processor used by the Mars 2020 rover has measured performance in the 200-300 MIPS range. In comparison, a 2016 Intel Core i7 measured over 300,000 MIPS, or over 1000 times faster. Furthermore, the Mars 2020 onboard scheduler is only allocated a portion of the computing cycles onboard the RAD750 resulting computation *several thousand times* slower than a typical laptop. While the Europa Lander Mission concept baselines considerably more computing than the Mars 2020 rover, it still likely to have far less computing than available on Earth.

technique shows significant improvement in utility achievement in the Europa Lander domain.

To demonstrate that the proposed approach is applicable to the execution of a realistic surface mission, we present a prototype implementation of onboard planning and execution software that exhibits these features (flexible execution and replanning with plan optimization) based on two primary software components: the Multi-mission EXECutive (MEXEC) [4] and the Europa Lander Autonomy Prototype (ELAP). We provide a detailed description of both components as well as our efforts towards integrating them into a prototype environment that models relevant behaviors of a hypothetical Europa Lander. Additionally, we demonstrate this planning and execution software in action commanding mission simulations under a wide variety of conditions representative of the wide range of situations a hypothetical Europa Lander might be expected to handle effectively.

The paper is therefore structured as follows: we begin with a description of the Europa Lander mission concept, including two concrete mission baselines, and our formulation of the problem as an HTN. Next, we describe the planning algorithm and the integrated planning and execution techniques – flexible execution and replanning with plan optimization – that comprise the main planning and execution approach. We then describe a prototype implementation of the proposed techniques using MEXEC, the Multi-mission EXECutive, integrated into a higher-fidelity simulation environment, the Europa Lander Autonomy Prototype (ELAP), to carry out realistic surface mission scenarios. Afterwards, we present empirical results of a basic surface mission simulation for the Europa Lander mission concept, and examine the effectiveness of our approach across a wide array of plausible mission scenarios. Finally, we conclude with a review of related work, and a brief discussion of future work.

II. Europa Lander Mission Concept

A. Problem Description

The primary goal of the Europa Lander mission concept is to collect samples of material from the Jovian moon's surface, analyze the sampled material for signs of biosignatures, and communicate that data back to Earth [1]. Because of the harsh radiation environment, sampling activities must be preceded by an excavation step, which removes the sterilized upper layers of the surface and exposes a region for sampling that is more promising in the search for biosignatures. Additionally, there are secondary objectives to take panoramic imagery of the Europan surface and collect seismographic data. Lander operations are generally limited to the accomplishment of these two overarching goals. This provides significant structure to the problem, since the mission concept clearly defines the sequence of actions required to achieve these goals.

The lander is tasked with excavating one or more sites on the Europan surface, collecting one or more samples from each site, analyzing those samples, and returning that data to Earth. The workspace accessible to the lander's robotic arm is divided into multiple excavation sites, and further into multiple sampling targets within each site. The relative scientific value of any given sample is unknown until it is collected, and the lander is expected to assess the potential value of each target in-situ using visual information prior to deciding which samples to collect and in what order. After excavation and sample collection, samples must be transferred into scientific instruments that analyze the material and produce data products. Data generated from sample analysis is assigned a scalar science value for planning purposes but, crucially, the mission is not considered to have achieved any actual utility from these sampling activities until the resulting data products have been communicated back to Earth.

In addition to sampling tasks, the lander may engage in seismographic data collection and period panoramic imagery tasks. These are considered secondary goals, with lower utility associated with their completion. As such, the data products that these tasks generate are considered to have lower value. However, these tasks also involve no surface interaction, and have less uncertainty associated with them as a result.

It is important to note that primary utility is only achieved when data is downlinked back to Earth.[2] This is true for both the sampling and seismograph/panorama tasks. Some excavation sites or sampling targets may provide more utility than others if, for example, one of those targets has a positive biosignature and the other does not. However, regardless of the quality of the material that the lander samples, no utility is achieved unless that data is communicated. This dynamic means that while potential utility is generated during the sampling and analysis phases, it is only realized by completing relevant communication tasks.

The Europa Lander mission concept is also constrained by a finite battery that cannot be recharged. Battery life is a depletable resource, and the lander must use its energy as efficiently as possible. Each task saps energy from the battery, and our algorithm must therefore plan accordingly to maximize utility given this constraint. In addition to this challenge, the surface characteristics of Europa are uncertain, and any prior mission model that is generated before landing is sure to have inaccuracies. In particular, the energy consumption of the excavation and sample collection tasks is largely unknown. There is also significant variation in the utility of any given sample, since the value of sampling a given target on Europa depends on whether the material is scientifically interesting, e.g. whether a biosignature is present.

B. Example Surface Missions

The Europa Lander team developed reference surface mission specifications to establish the context and scenarios which Europa Lander surface mission autonomy would be expected to operate within [1]. These were used to clarify the set of functionalities that a Europa Lander Autonomy Prototype would need to implement in order to simulate a surface mission, as well as to provide a framework for designing for the scenarios that should be run on the completed software. We describe these below.

[2]While context imagery while preparing sampling locations has some science value, the primary science value is from analysis of the samples.

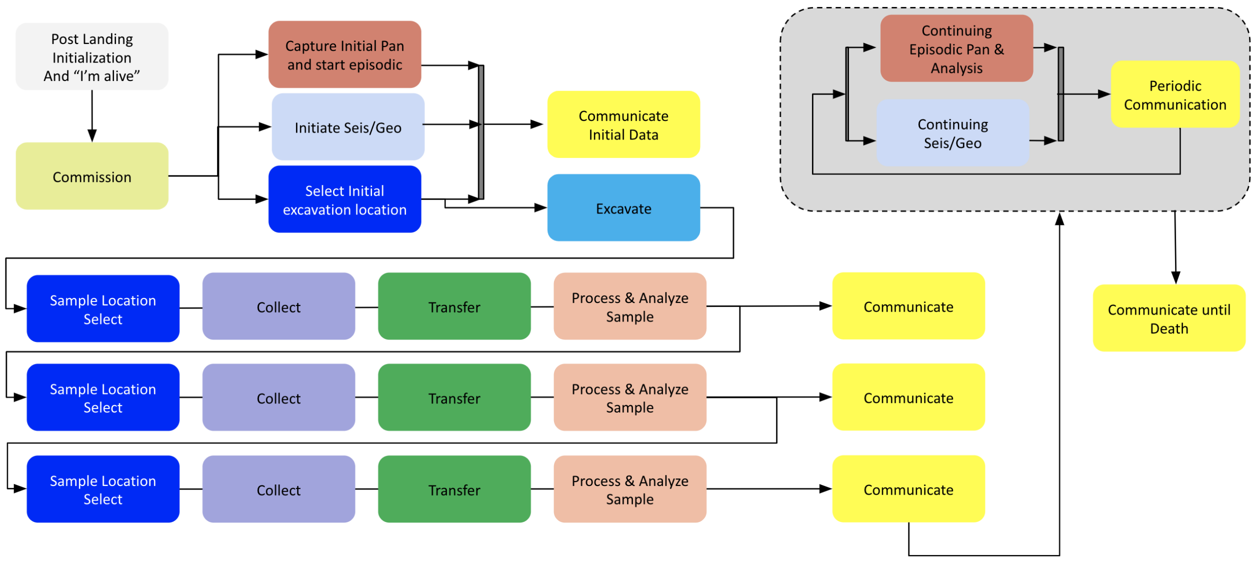


Fig. 1 Sampling Process Specification for Basic Surface Mission 1 (BSM-1)

1. Basic Surface Mission

Basic Surface Mission 1 (BSM-1), depicted in Figure 1, was used to guide initial prototype development, and describes a deliberately minimal set of requirements for surface operations that includes excavating a single site, collecting three samples from that site, and downlinking the resulting data. In parallel to sampling, the lander must also command ongoing seismometry and periodic panoramic imaging tasks. This reference mission includes no "ground-in-the-loop" interactions and is to be executed entirely autonomously. In BSM-1, the science values assigned to data generated from sample analysis are fixed values assigned to each potential sample, but unknown to the system at the start of the mission and computed onboard. Figure 1 displays the strong dependency structure inherent to the Europa Lander mission concept. In order to sample, the lander needs to have excavated a site; in order to analyze, the lander needs to have collected a sample; etc.

2. Reference Surface Mission

Reference Surface Mission 1 (RSM-1), depicted in Figure 2, is designed to explore the complexities introduced by ground-in-the-loop (GITL) management of initial excavation and sampling activities. In RSM-1, the lander is tasked with collecting, analyzing, and communicating results for up to three samples at each of a number of different candidate excavation sites. Depending on the amount of energy available to the lander and the resource use of onboard activities, it may not be possible to return data for all reachable sampling targets, so the system must intelligently prioritize activities to maximize productivity. The science values in RSM-1 for the analysis data are assigned as a function of the full history of the biosignature results from any already-analyzed samples, along with a weighting of the estimated relative value of each remaining sample target as determined by visual inspection.

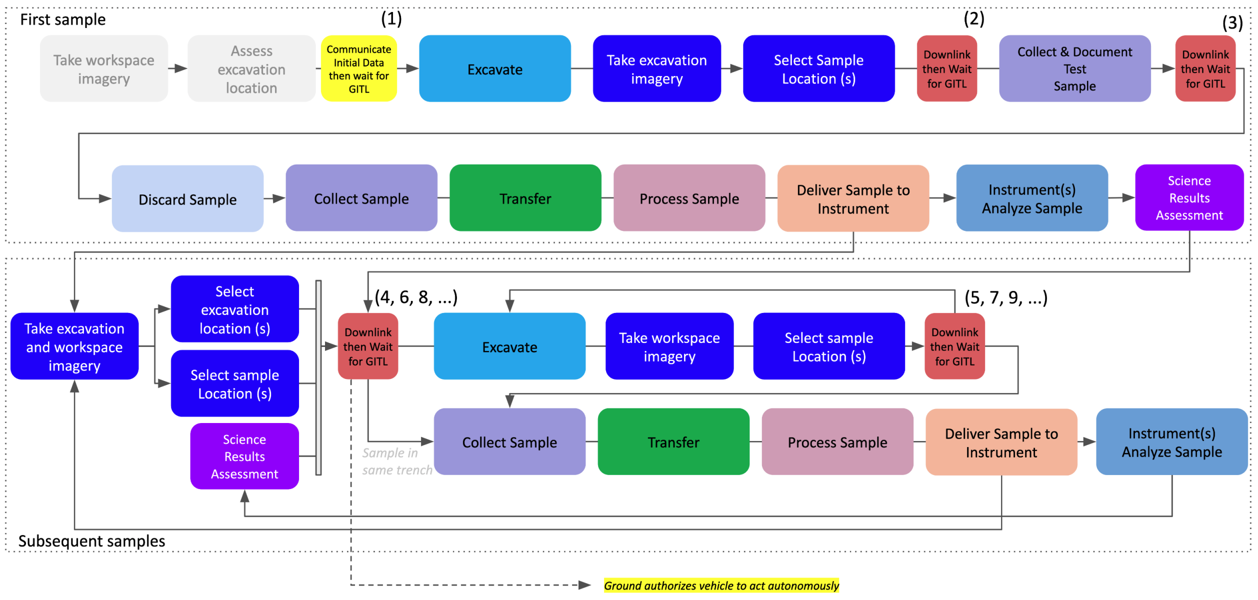

First sample
Take workspace imagery
Assess excavation location
(1)
Communicate Initial Data then wait for GITL
Excavate
Take excavation imagery
Select Sample Location (s)
(2)
Downlink then Wait for GITL
Collect & Document Test Sample
(3)
Downlink then Wait for GITL
Discard Sample
Collect Sample
Transfer
Process Sample
Deliver Sample to Instrument
Instrument(s) Analyze Sample
Science Results Assessment
Take excavation and workspace imagery
Select excavation location (s)
Select Sample Location (s)
Science Results Assessment
(4, 6, 8, ...)
Downlink then Wait for GITL
Excavate
Take workspace imagery
Select sample Location (s)
(5, 7, 9, ...)
Downlink then Wait for GITL
Sample in same trench
Collect Sample
Transfer
Process Sample
Deliver Sample to Instrument
Instrument(s) Analyze Sample
Subsequent samples
Ground authorizes vehicle to act autonomously

Fig. 2 Sampling Process Specification for Reference Surface Mission 1 (RSM-1)

There are checkpoints within the task specification after which a “ground hold” flag is applied, and sampling may only progress after ground operators release the hold on a subsequent uplink. This creates opportunities for ground operators to inspect and influence the onboard decision-making process if desired. A “fully autonomous” flag is also provided which, when set, removes these ground-in-the-loop checkpoints and allows the lander to plan and command multiple sampling cycles without intervention.

This scenario also introduces four tiers of data products: *Critical* data, which must be transmitted immediately, *Decisional* data, which must be transmitted at the next pre-determined downlink time to provide context for ground operators, *Mandatory* data, which must be transmitted before the end of the mission, and *Residual* data, which is only to be transmitted if additional energy remains at the end of the mission. Finally, this reference mission introduces configurable limits on daily and per-downlink energy allowance. The sampling flow for RSM-1 is illustrated in Figure 2.

The behaviors illustrated in these diagrams represent commands that can be issued by the Planning and Execution component. Delineating appropriate interface boundaries was one of the important questions in the early design stages of the autonomy prototype and the reference missions.

In RSM-1, the policy that the lander should follow is to collect up to three samples from a site, continuing to sample at that site as long as positive biosignatures are discovered there, and moving on to the next site whenever a negative biosignature is discovered. It should choose sites, and samples within a site, in the order of relative estimated science value as determined by onboard vision algorithms run on imagery of the lander’s workspace. We refer to this valuation scheme as as the *science subway map* and it is depicted in Figure 3.

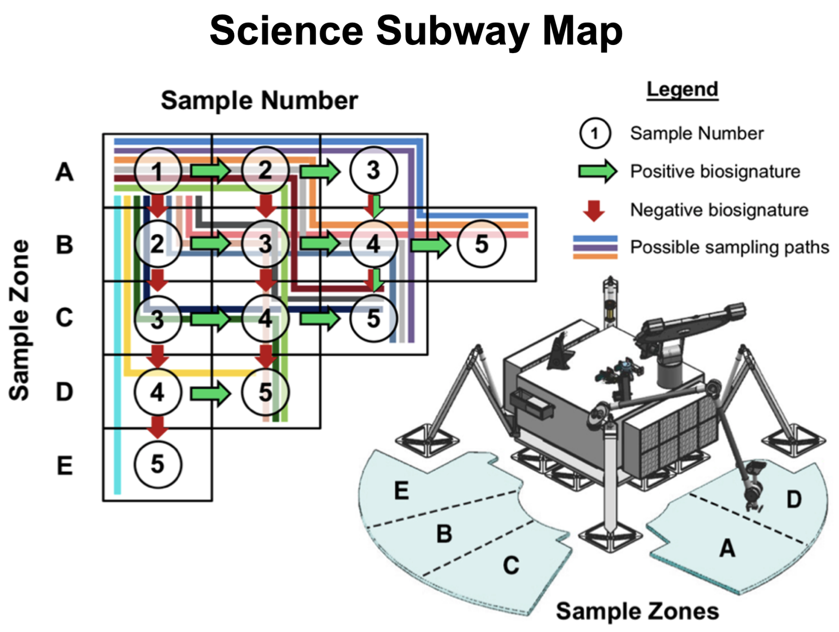


Fig. 3 "Subway Map" Sample Selection Strategy In Reference Surface Mission 1 (RSM-1)

C. Problem Formulation

We model the problem of commanding the Europa Lander reference missions using hierarchical task networks (HTNs).While other decision-theoretic models for sequential decision-making, such as Markov decision processes, have been used successfully in many settings, they are computational expensive and generally not well-suited for domains with concurrent actions and continuous states, as considered here. HTNs, however, allow us to compile significant domain-specific knowledge of the dependency structure across tasks into the task network to improve performance so that it can be solved onboard. Indeed HTNs have been used successfully in industrial and other real-world applications to improve the tractability of planning problems in systems such as SHOP2 [5] and SHOP3 [6].

In an HTN, hierarchical tasks are decomposed into a set of subtasks. We refer to the higher-level tasks as "parent tasks", and refer to their children as "subtasks". Parent tasks may decompose into a number of different sets of subtasks; we refer to each of these sets as a potential "decomposition" of that parent task. Finally, we refer to tasks with no decompositions as "primitive tasks". These primitive tasks represent tasks that the lander can be directly commanded to perform. In the RSM-1 mission concept, the primitive tasks that we modeled are:

- `Take Mono Panorama`
- `Find Excavation Sites`
- `Prepare Arm`
- `Excavate Site(N)`
- `Find Collection Targets(N)`
- `Collect Sample(N, M)`
- `Transport Sample`
- `Prepare HGA Comm`
- `Discard Sample`
- `Start Seismometer`
- `Stop Seismometer`
- `Science Analyze`
- `Downlink`
- `Downlink with Ground Hold`
- `Uplink`

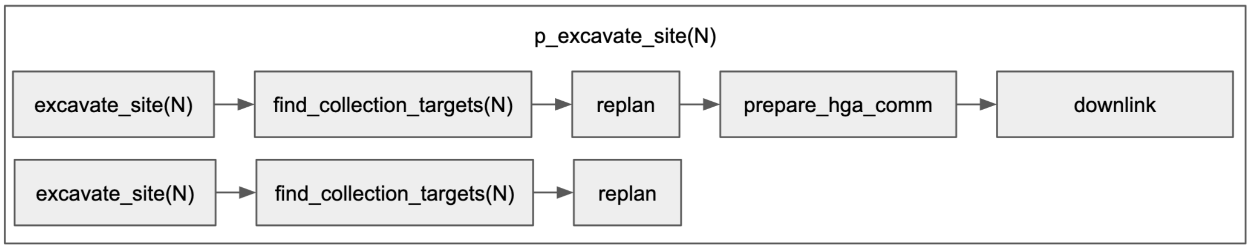


Fig. 4 Two possible decompositions of a single parent task; which of these can be scheduled depends on the current mission phase, which is explicitly modeled on a timeline.

In addition to these commandable tasks, we also define a `Hotel Load` that is scheduled at the start of the mission and models a small continual energy cost and a `Seismometer-ON` task which represents the resource costs associated with running the seiesmometer instrument.

Decompositions enable us to compile domain-specific knowledge of task-dependencies into the the task network, significantly reducing the plan search space. In addition, we can treat all subtasks of a parent task as a singular block for planning purposes. The lander primarily achieves utility after completing an entire sequence of sample, analyze, communicate. Decompositions allow us to treat "sample, analyze, communicate" as a single unit and schedule them accordingly. Thus, our model intrinsically biases the lander against planning to sample without a corresponding communication task. This may not always be optimal; for example, when excavation and sampling is cheap and communication is very expensive. However, for our problem, energy use is dominated by the excavation and sampling tasks, and the decomposition paradigm effectively encodes this domain-specific knowledge into our planning routine.

There are three main parent task types in our mission model. The first is a Preamble task, which consists of post-landing initialization and other one-time initialization tasks. Second are sampling tasks, which consist of excavation, sample collection, transfer, analysis, and communication sub-tasks. Excavation can take place at one of three excavation sites, and may be skipped if an excavation has previously occurred for the specified site. For collection tasks, the lander may choose between nine collection targets – three for each excavation site Then, for communication tasks, the lander may choose to either communicate raw data or compressed data. Finally, there are Seismograph/Panorama tasks, which consist of seismographic data collection, panoramic image collection, and communication of that data.

We assign utility primarily to two activities: sampling and communication. Both of these task models are assigned a numeric value representing their utility, which can be updated online by the planning and execution system if knowledge at execution time alters the expected utility of a given action. Utility for these tasks is achieved only after their full decomposition has been successfully executed. Thus, for sampling utility to be achieved, a corresponding communication step must successfully complete.

We assign utility to sampling tasks in order to differentiate between sites that may be more or less interesting, depending on the scientific value of the site. Communication utility is larger, and remains constant. For the communication tasks, we assign higher utility and cost to tasks that communicate raw data, compared to those that communicate compressed data. This simulates a Pareto optimal "menu" of communication options. The combination of sampling and communication utilities represents the overall utility of a parent sampling task. Seismograph/panorama utility is driven solely by communication utility.

Formally, we define our planning problem as follows. An input to the problem is a set of tasks $\mathcal{T} = \{t_1, ..., t_n\}$, with each task t_k represented by a tuple $\{c_k, u_k, d_k, P, I\}$ where:

- c_k represents the task's cost,
- u_k represents the task's utility,
- d_k represents the task's nominal duration,
- P is the set of the task's precondition (these may be based on resource values, or on the execution state of dependency tasks), and
- I is the set of its impacts on resource timelines.

This matches the timeline representation of execution state used by [7]. For our problem, we assume that we have a fixed cost budget b. In the Europa Lander domain, this budget represents the non-rechargeable battery, with each task using up some amount of that battery's energy. We therefore aim to maximize utility by scheduling tasks subject to the following constraints:

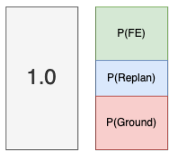
- For all tasks, all preconditions are valid.
- For all tasks, all impacts are valid.
- The sum of all task costs does not exceed b.

Given this context, we predict the overall utility achievement of a plan using an estimate of utility per unit cost $u_{avg} = \sum_k \frac{u_k}{c_k}$. If every task in the plan is always successful, our expected utility for a plan under this prior would therefore be bu_{avg}, the maximum utility achieved under that plan with cost-budget b. To factor in task failure, we assume that tasks fail with a fixed probability $P(\text{fail})$ following a Poisson distribution, so that each task fails with a fixed failure rate,[3] and consider three increasingly costly methods for failure recovery: *Flexible Execution*, *Replan*, and *Ground*. Generally, the cost of resolving failures is in the form of time and energy which may reduce the opportunity for future task execution, and hence potential future reward. We describe these methods in detail in the next section, but note here that we assume that some subset of these failures can be resolved with flexible execution, a strictly larger subset can be resolved using replanning, and the largest set of failures can be resolved via waiting for ground input.

Consequently, if $P(\text{FE})$ is the probability that a failure is resolvable by flexible execution, $P(\text{Replan})$ is the probability that an event is not resolvable by FE but is resolvable by Replan, and $P(\text{Ground})$ is the probability that an event is not resolvable by either FE or Replan but is resolvable by Ground, we find that

[3]Extending the analysis to more realistic plan structure, failure patterns, and interactions with exogenous events are future work.

$$P(\text{failure that occurs is resolved}) = P(\text{Ground}) + P(\text{Replan}) + P(\text{FE}). \tag{1}$$



In practice we assume that $P(\text{Ground}) = 1.0 - P(\text{Replan}) - P(\text{FE})$, i.e. waiting for Ground can resolve *any* failure mode that is not resolved by FE or Replan, which brings the probability of a failure occurring and being unresolved to 0.0 (see figure above), but may be costly to perform as Ground is only intermittently available.

III. Approach to Planning and Execution

In this section we describe both the main planning algorithm and the on-board autonomy methods that comprise the overall planning and execution approach proposed.

A. Utility-Maximizing Search

Our planning algorithm uses the HTN model of the Europa Lander problem to build a search graph, with nodes holding partial plans and edges holding task decompositions. We perform a heuristic-guided constrained-search on this graph and select the best plan explored. The algorithm consists of four phases: pre-processing, initialization, exploration, and plan selection.

Pre-Processing

First, a pre-processing step flattens task decompositions into a single layer, such that parent tasks decompose into a chain consisting only of primitive, non-hierarchical subtasks (see, for example, the task chains for sampling presented in Figure 4). This allows us to assign utility and energy cost directly to each decomposition, because its breakdown into disparate subtasks has already been performed. Then, each decomposition's expected utility is the sum of each of its subtasks' utility, and the same is true for energy cost. This step is performed offline one time per domain model. While this preprocessing has exponential runtime in the worst case, for our Europa Lander (as well as most space applications we have seen) the majority of the search occurs in scheduling the expanded tasks, not action selection, so this preprocessing is tractable (managing this complexity is an area for future work).

Initialization

Our search tree consists of nodes containing partial plans and their associated energy cost and utility. A node's cost is the sum of the costs of each task in the node's partial plan; and likewise the node utility is the sum of the utilities of the (sub) tasks (task utility that is dependent on other tasks is a topic for future work). In the initialization phase, the algorithm creates a single node containing an empty plan, with utility and cost 0. Then, it iterates through all

Algorithm 1: Europa Lander Planning

```
Input: A list of tasks to schedule $T$
Output: A plan of scheduled tasks $P$
/* initialize exploration queue */
node_collection = [];
add (plan=[], utility=0, cost=0) to node_collection;
edge_collection = [];
for d in task.decompositions do
    new_edge = $(d, d.utility, d.cost)$;
    add new_edge to edge_collection;
end
explore_q = [];
for edge in edge_collection do
    add (node_collection[0], edge) to explore_q;
end
/* search exploration queue */
num_explored = 0;
while num_explored below exploration bound do
    num_explored++;
    plan, decomp = explore_q.get_max();
    if decomp tasks can be added to plan then
        new_plan = plan + decomp tasks;
        add new_plan to node_collection;
        for edge in edge_collection do
            edge.cost_total = edge.cost + compute_heuristic(edge);
            if edge.task not in new_plan and new_plan.cost + edge.cost_total below max_cost then
                add (new_plan, edge) to explore_q;
            else
                prune(edge);
            end
        end
    end
end
/* find best plan in node collection */
best_plan = null;
for plan in node_collection do
    if plan.utility above best_plan.utility then
        best_plan = plan;
    end
end
return best_plan;
```

task decompositions created in the pre-processing phase in order to generate the set of edges that may be followed from a given node. To finish the initialization phase, the algorithm populates an exploration queue with (node, edge) pairs, pairing the singular initial node with all edges in the collection. At the end of the initialization phase, then, the exploration queue consists of all task decompositions paired with the empty plan.

Exploration

In the exploration phase, the planner pops the top of the exploration queue to get (P, T), where P is a partial plan, and T is the list of primitive subtasks comprising a task decomposition. It then attempts to schedule all tasks in T given the state of the world produced by following the plan P. If the tasks cannot be scheduled, it moves on to the next exploration queue item. If the tasks can be scheduled, i.e. their preconditions are met and their impacts do not produce any conflicts, a new graph node is created. This node contains a new plan P', the resulting plan after adding the tasks in T to P.

Plan Selection

After creating this plan node, the planner iterates through the edge collection again, pairing the new plan with all possible tasks. In this iteration, it ignores tasks that have already been scheduled in the plan, so as to avoid duplicates. The algorithm also filters these pairs to ensure that the total cost $P.cost + T.cost < M$, where M is the max energy cost allowed (equal to the current battery charge of the lander). This bounds our search by pruning from our search space the entire subtree that the edge leads too, and we further bound the algorithm's search by limiting the number of exploration candidates examined. Note however that this bound maintains optimality if we allow the algorithm to expand the entire space, as any subtree pruned due to infeasibility must necessarily have utility no greater than the best-found plan at the time of pruning. After filtering, these pairs are added to the exploration queue, and the next queue item is examined. The exploration queue is a priority queue, with (plan, decomposition) pairs ordered by a heuristic value to improve search results. Given a plan, decomposition pair (P, T), we assign the heuristic value $h(P, T) = P.utility + \frac{T.utility}{T.cost}$. Finally, in the plan selection phase, the algorithm iterates through all candidate plan nodes, selecting the plan with the highest utility. Ties are broken according to energy cost, where lower is preferred.

In the RSM-1 implementation, we add an outer loop that iteratively builds the schedule in priority-ordered phases. First, fixed exogenous uplinks are added as the highest priority. Next, prioritized batches of tasks representing phases of the mission are taken as candidates for scheduling; each time a best schedule is found with a given set of tasks, that schedule is held fixed and used as the initial node in scheduling the next lower priority batch of tasks. Scheduling independent phases of the mission in this manner allows us to further reduce the search space.

B. Flexible Execution

Flexible Execution (FE) is a lightweight rescheduling algorithm that runs at a much higher cadence than the planner [8, 9]. *FE* has two main properties: (1) it is much less computationally demanding than replanning as it does not search, and (2) it is less capable than replanning. *FE* allows the system to handle less-severe unexpected events without incurring the cost of replanning. Previous NASA missions have made heavy use of flexible execution, such as the Mars 2020 Perseverance rover [10]. Our implementation differs in focus, emphasizing responses to adverse events.

In our system, *Flexible Execution* consists of two major components. The first is `task push`. If a task's preconditions are not met, before considering the task to have failed to execute, we allow the system to wait for some amount of time for this inconsistency to resolve. Such a situation might occur, for example, if required preceding activities are delayed or run long. The executive checks the task's preconditions and delays dispatch until either the conditions have been met, or the task's wait timeout has been exceeded. The second component of *Flexible Execution* is `automated retry`. If a task completes with a failure code, *FE* can immediately re-schedule the task if its preconditions are still met (and update the plan with the new predicted end time and resource usage of the task), avoiding replanning cost and delay.

In the context of the Europa Lander domain, *Flexible Execution* offers significant value because many robotic tasks such as excavation and sample acquisition can vary significantly in duration and hence resource consumption. Moreover, the algorithm handles this variation without disrupting the execution flow.

C. Replanning with Plan Optimization

For more complex execution variations, we turn to replanning during execution. Replanning uses search to construct the plan based on the current state. The current state includes: state and resource values (accounting for failed activities, and resource under/over runs) as well as the current time (accounting for execution time variation). Additionally, utility estimation, predicted duration, and predicted resource expenditures for future tasks may be updated (e.g. imaging may indicate that a sampling site now looks more promising and/or may take longer/shorter to excavate or sample). Replanning enables incorporation of all of this new information into decision-making.

D. Planning and Execution Strategies

Using these methods, we examine four planning and execution strategies of increasing onboard autonomy: `Static`, `Ground`, `Flexible Execution (FE)`, and `Replan`.[4]

In `Static`, a plan is generated before execution time, then executed without change. No failure responses are available, so the first task failure results in the remainder of the plan not executing. In `Ground`, we introduce a mechanism for failure resolution: waiting for ground input. We assume that ground input is able to resolve all failures. The plan generated on the ground, and any task failures result in a halt to plan execution, ground fixes the problem, then execution resumes, albeit incurring considerable cost. In the `FE` strategy, we allow flexible execution of our plans using the *FE* algorithm, which can resolve some but not all failures, with all other failures handled by waiting for ground input. Finally, in the `Replan` strategy, `FE` is applied first to attempt to resolve failures; if `FE` cannot resolve the failure, replanning is applied; if replanning cannot resolve the failure, the ground strategy is applied (See Fig 5).

We describe each of these strategies below.

[4]Note that in each case, the plans are generated using the Uitility-Maximizing Search algorithm.

1. Static Strategy

We first analyze the `Static` strategy. Assuming the plan is a linear sequence of activities and execution terminates with the first failure, the number of tasks executed before the first failed task follows a geometric distribution with probability P(fail). Hence, the expected number of attempted tasks needed to observe a failure is $1/$P(fail), and the expected number of successfully executed activities is (1/P(fail) - 1) (but bounded by the number of tasks n). Therefore the static expected utility is the lower of the "no failure case" (at left below) or (at right) the expected utility per task $u_{avg} \cdot c_{avg}$ times the number of expected activities executed as indicated below:

$$U(S_s) = \min\left(u_{avg} \cdot b, \frac{u_{avg} \cdot c_{avg}(1 - P(\text{fail}))}{P(\text{fail})}\right) \tag{2}$$

where c_{avg} denotes the average cost of each task.

2. Ground Strategy

In the `Ground` strategy, the only error response for all failures is to "go to ground" to resolve them, which always allows plan execution to continue. If our plan has n tasks, this occurs $nP(\text{fail})$ times costing c_g energy each occurrence. The utility achievement of this strategy is therefore:

$$U(S_g) = u_{avg}\left(b - P(\text{fail})nc_g\right). \tag{3}$$

3. Flexible Execution Strategy

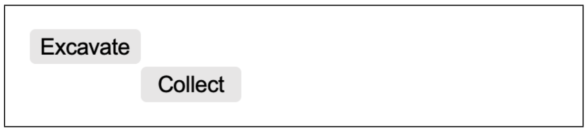
In the `FE` strategy, we introduce *Flexible Execution* and assume that some subset of task failures can be resolved with this feature. The probability of a task failing in this way is $P(\text{FE})$ and note that $P(\text{FE})$ is a subset of the total failure cases $P(\text{fail})$. We assume that *Flexible Execution* has a negligible cost. Then, the utility achievement of plan execution using this strategy is:

$$U(S_f) = u_{avg}\left(b - (P(\text{fail}) - P(\text{FE}))nc_g\right). \tag{4}$$

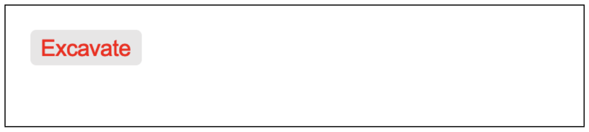
4. Replan Strategy

Finally, we consider the `Replan` strategy, which incorporates flexible execution and replanning with plan optimization. Unlike flexible execution, replanning incurs some non-negligible cost c_r. We assume that, like flexible execution, replanning is able to resolve some subset of task failures. As noted above, we denote the probability that a given task fails in a way that can be resolved via replanning, but not flexible execution, as $P(\text{Replan})$.

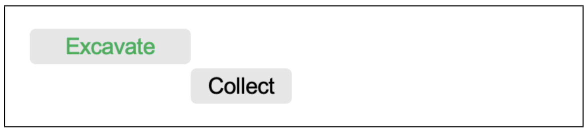
Failures are resolved by the least costly resolution mechanism. Thus, when a task fails, our system attempts to resolve it by flexible execution, if possible, falling back to replanning and ground intervention in sequence. To model plan optimization, we provide our planning system with opportunities to discover utility at certain points during execution.



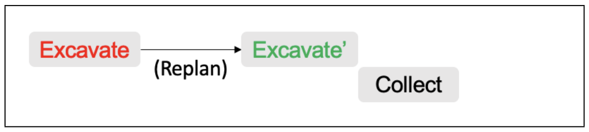
(a) Original Plan



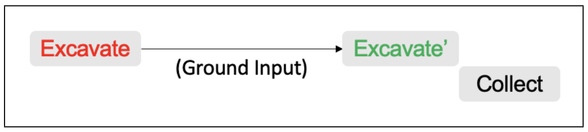
(b) Static Strategy: task failure leads to early termination



(c) Flexible Execution: task extended to continue execution



(d) Replanning: task replaced after brief replanning period



(e) Ground Intervention: task replaced after longer ground communication period

1. **Black text indicates planned future tasks.**
2. **Red text indicates failed tasks.**
3. **Green text indicates successfully executed tasks.**

Fig. 5 Execution Strategies

We denote the number of such opportunities as d, and the expected additional utility discovered as u_d. Then,

$$U(S_r) = du_d + u_{avg}\Big(b - n(P(\text{Ground})c_g + P(\text{Replan})c_r)\Big). \tag{5}$$

A description of the utility discovery mechanism can be found in Section IV.C.2.

IV. Simulating On-Board Autonomy for the Europa Lander

As the Europa Lander mission concept provided both the justification and motivation for the work discussed in this paper, a critical component of validating the effectiveness of our approach is to have a realistic simulation of the domain. There are two main software components that comprise the simulated set-up: *Multi-Mission Executive* (MEXEC), the onboard planning-and execution software, and *Europa Lander Autonomy Prototype* (ELAP), which simulates the execution of realistic surface mission scenarios. We describe both components, and their integration, below.

A. MEXEC

For our empirical evaluation and simulation work we use MEXEC [4] to implement the approach to autonomy we have presented here. MEXEC (Multi-mission EXECutive) is an onboard planning and execution software that uses task networks to generate and execute conflict-free schedules to achieve goals. It consists of three modules: a *planner*, an *executive* (also sometimes referred to as a *controller*), and a *timeline library*, the latter of which is used by the planner to search for valid intervals to place tasks. MEXEC was designed specifically to be used for flight software and to have a consistent, cooperative design between the planner and executive.

Figure 6 illustrates the MEXEC modules and their interactions with the ground and other flight components. MEXEC assumes that there is a state database that reports system state and a command dispatcher that dispatches commands. The ground sends a task network to the spacecraft, which is read by the planner. The planner schedules the tasks with the help of the timeline library and the system state from the state database. Once sufficiently close to the start time of a task, the planner passes it on to the executive, which performs real-time constraint checking to ensure safe execution of the task. Task execution updates are sent from the executive to the planner to keep the planner informed in case re-planning is necessary.

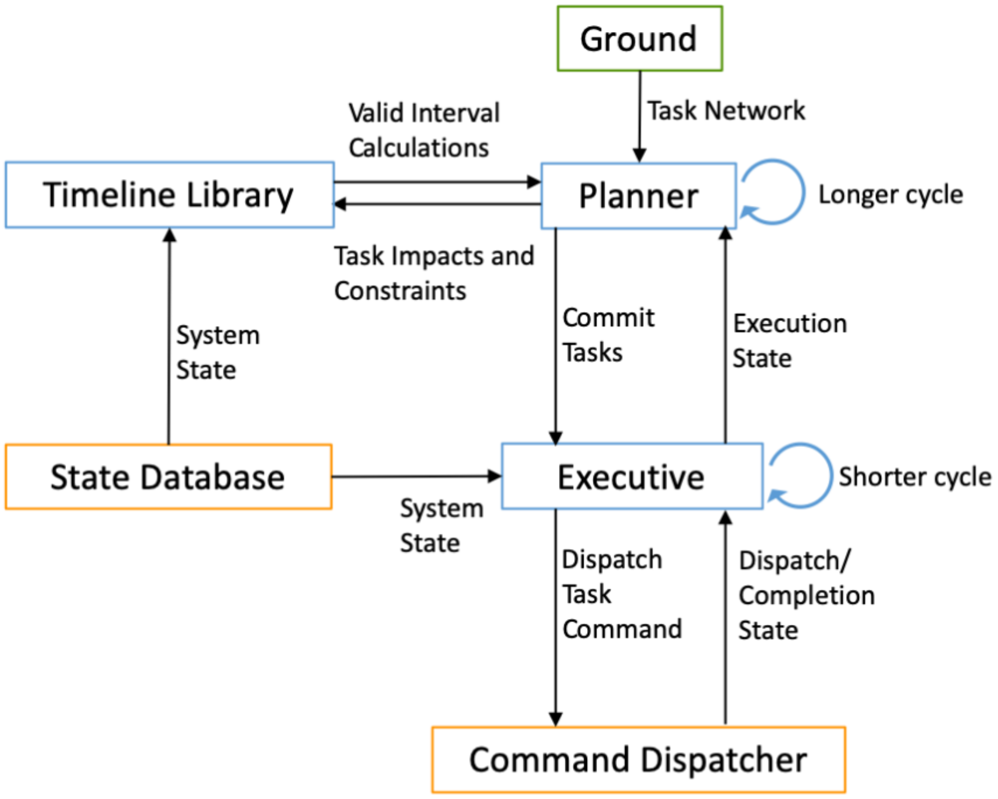


Fig. 6 Diagram of the interactions between components in MEXEC.

1. Task Network

In the context of MEXEC, a *task network* is an abstract representation of the behaviors that the executive has authority to command, the constraints on timing, precedence, and resource availability that must be obeyed when executing those behaviors, and the expected resource use of those behaviors, grouped to achieve some goal or to cover some temporal extent.

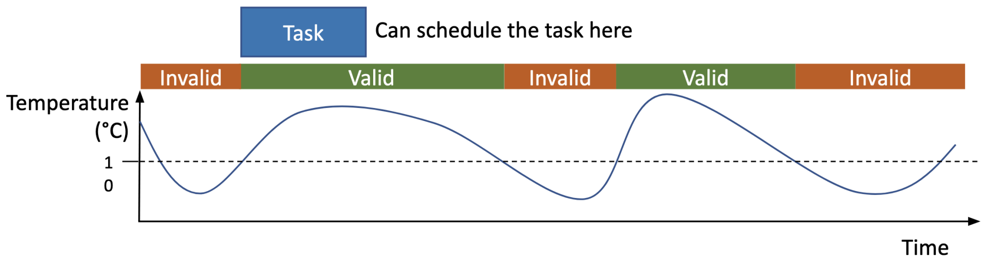


Fig. 7 Both past measured values and future projected values of a resource are captured on a timeline. Constraints on the timeline value are used to determine valid intervals for scheduling a task. Image credit [4].

Tasks represent a desired change to the system. Besides a unique ID and a name to identify them and a ground-specified priority to inform scheduling, tasks include a command to execute (which may be NO-OP or even a sequence) as well as expected effects of executing the task based on spacecraft behavior (impacts) and the conditions required for execution (constraints). Impacts and constraints can be defined at the beginning (pre), during (maintenance), or at the end (post) of a task. Conditions to set limits on task deviation at execution time, such as timeouts for waiting on constraints to be met or when to skip tasks, can be defined as control conditions. In the case of execution failure, tasks can also specify contingencies, which are actions to take on the task network by the planner, such as adding new tasks from templates. Templates are tasks that are available to the planner, but that are not explicitly requested to be included in the schedule. An immediate response by the executive in the case of task failure can also be defined in the form of a command to execute.

2. *Timelines*

MEXEC uses *timelines* to model its predictions of the values that system resources and states will take as a function time. These timelines are used by the planner to reason about task feasibility and timing, as illustrated in Figure 7. Timeline definitions include a type (which allows for modeling continuous vs discrete resources), an initial value, and a specification of the range of possible values that the timeline can take. When a task that includes one or more timeline impacts is placed by the planner, its impacts are applied to the relevant timelines at the relevant times, and in that way the timeline comes to represent the expected value of a given resource at any point in time during the mission as a function of the behaviors that MEXEC plans to command.

The timeline library supports planning by calculating valid intervals for tasks. Many of the concepts outlined in [11] for common capabilities for different timeline representations are supported by this timeline library, including support for analyzing schedules and constraints for state and resource timelines, which we separate into atomic, state, claimable, cumulative and cumulative rate timelines. Each resource or state that is referred to in a constraint or impact is represented as its own timeline and can either portray values reported by the spacecraft system or values managed internally by MEXEC, also known as internal states. Impacts from tasks are placed on the timelines to change the timeline value and are projected into the future to predict future state to compare to constraints and determine conflicts. Impacts can assign a value, a change in value, or a change in rate to a timeline. The aggregation of impacts and their projections provide a timeline result, or expected value, at any given time. A constraint is in conflict if the result during the constraint is not within the prescribed values. System state updates from the state database are also applied to the timelines to keep the latest spacecraft values synchronized. This allows the planner to perform its periodic constraint checking on all states with up-to-date values.

During scheduling, the planner uses the timeline library to look for valid intervals in which to place a task. A valid interval is one in which a task can be placed without creating conflicts. Any previously scheduled tasks have their impacts and constraints placed and projected on the relevant timelines. To find valid intervals for a task, the timeline library systematically places the task at times within its commit window and checks for conflicts. Timeline results are also adjusted when feedback from the executive notifies the planner that task execution has deviated from its originally scheduled time. In this case, the task's impacts and constraints are moved to reflect the actual execution of the task.

Additional information about the timeline library can be found in [9].

3. Planner

The MEXEC planner is a component that takes as its input a task network and, using the valid interval calculations provided by the timeline library, generates a conflict-free schedule of tasks to perform. It then commits tasks at the appropriate time to the controller to be commanded. Figure 8 shows an example visualization of a schedule output by the MEXEC planner.

The planner, with the valid interval calculations provided by the timeline library, generates and maintains conflict-free schedules given an input task network. Every planning cycle, the planner takes the following actions in order:

1) Commit tasks to the executive
2) Consolidate impacts
3) Schedule tasks (configurable)
4) Repair conflicts (optional)
5) Optimize the schedule (optional)

The planner makes use of various timing windows to inform decisions. The first is the plan process interval, which defines the frequency at which the planner cycle is run. This interval must be longer than the worst case duration of the planner cycle. The plan process interval also influences the duration of the commit window, which is used to determine which tasks should be passed on, or committed, to the executive, first defined in [12]. Any scheduled task with a start time before the end of the commit window should be committed. Since tasks are only committed at the beginning of the planning cycle and tasks should be committed before their start time, the commit window must at a minimum be as long as the plan process interval.

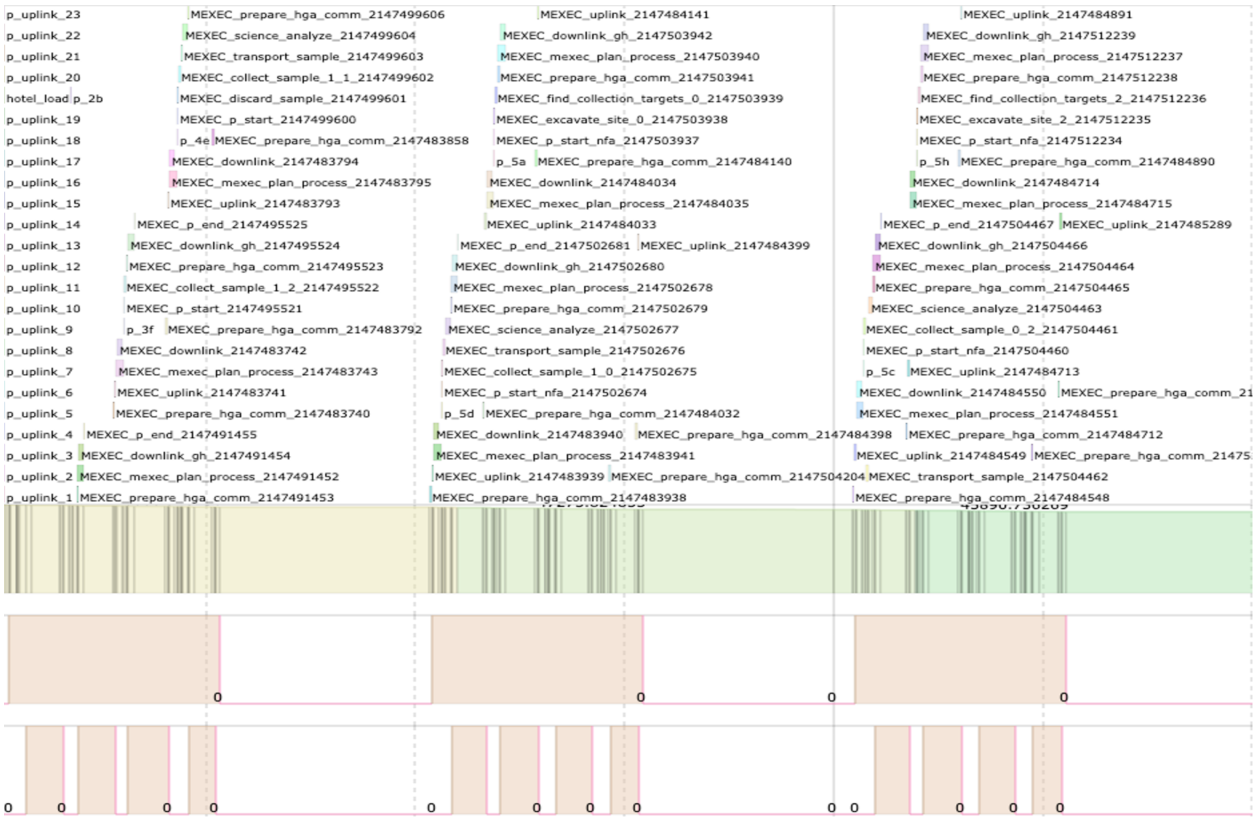


Fig. 8 Sample of a full mission schedule: 3 sites excavated, 9 samples collected over multiple sols, plus episodic imagery, fixed exogenous uplink windows, downlinks. Mission ends when battery depleted.

Another important window is the scheduling window. Unscheduled tasks with a preferred start time before the end of the scheduling window are considered for scheduling by the planner during the planning cycle, allowing the planner to work on a subset of the problem at a time. The planner then schedules the tasks within the plan horizon, which starts after the commit window, to prevent tasks from being scheduled before they can be committed, and extends to the plan end time, which defines the time before which all tasks in the task network should be scheduled. During scheduling, the planner checks for conflicts during the conflict checking window, which starts after the commit window and goes until the maximum time allowed on timelines. The conflict checking window starts after the commit window since any tasks already committed to the executive should not be changed by the planner since they may have started executing already, and any conflicts that occur at execution time will be handled by the executive.

After committing tasks, the planner consolidates impacts on the timelines, combining the past impact results into a single result to reduce the memory necessary to represent a timeline. The memory footprint of MEXEC in general is adjustable based on various configuration values. Next, the planner schedules any tasks in the scheduling window. One feature of MEXEC is that it is easy to plug in different scheduling algorithms that are most appropriate for the application in which it is being used.

The planner also handles incoming information when it is not executing its repeated cycle. This includes reading task network files, handling state updates from the state database to be placed on timelines (as mentioned in the Timelines section), and interpreting task execution updates from the executive. Task networks are uploaded to the spacecraft from the ground in the form of a binary file. The planner reads the file and performs input validation before storing the information to ensure that the incoming timelines and tasks will fit within the available memory and satisfy any assumptions made by the tasks. Both state database updates and task updates are used by the planner to update timelines, which provide information to the planner in the case that re-planning is required. Task updates also notify the planner of task completion status, including if and why a task failed. This information is used to determine if any contingency actions need to be taken.

4. Executive

The executive is the component responsible for task execution and handles adjustments needed for execution deviation (i.e. flexible execution). The MEXEC executive converts constraints and control conditions into boolean expression trees to be evaluated at each stage of task execution on a per task basis. The evaluation is based on system state reported from the state database or from internal state propagation. Each transition to a new stage and any faults are reported to the planner.

In the executive, all constraints are monitored separately for each task. Pre-constraints, including the start time of the task, must be achieved before the task will start. If the constraints are not achieved within a defined timeout interval, the task is considered failed. Once the constraints are achieved, the executive dispatches the command associated with the task and starts monitoring any maintenance constraints in the task. The executive expects a command response to indicate that the command was dispatched and completed successfully. If the command response comes back with an error, or the command response does not come back within a defined timeout interval, the task fails. Additionally, if any of the maintenance constraints do not hold true before the end of the task, the task fails. If the command response returns with success, and the maintenance constraints hold, the task completes with success after the duration of the task has been reached.

B. Europa Lander Autonomy Prototype (ELAP)

To demonstrate the effectiveness of our proposed approach to system-level autonomy, we implemented a Planning & Execution module in the context of the Europa Landed Autonomy Prototype (ELAP), a suite of software developed to simulate the execution of realistic surface mission scenarios [13]. The purpose of ELAP is to provide a platform for rapid scenario experimentation to investigate different surface mission autonomy operations concepts, technologies, and architectures for a Europa Lander mission. A major focus of the prototype was to provide a platform for testing the ability of planners and executives to successfully carry out mission scenarios, including responding to variations in the state and behavior of the system and its environment and transmit data back to the Earth.[5]

ELAP consists of a modular set of components, which can be broadly categorized into: simulations of the environment external to the lander, simulations of the lander itself (e.g. hardware and instruments), prototype implementations of onboard flight software, prototype implementations of ground systems, and user interface elements. The relationships between these modules are illustrated in Figure 9.

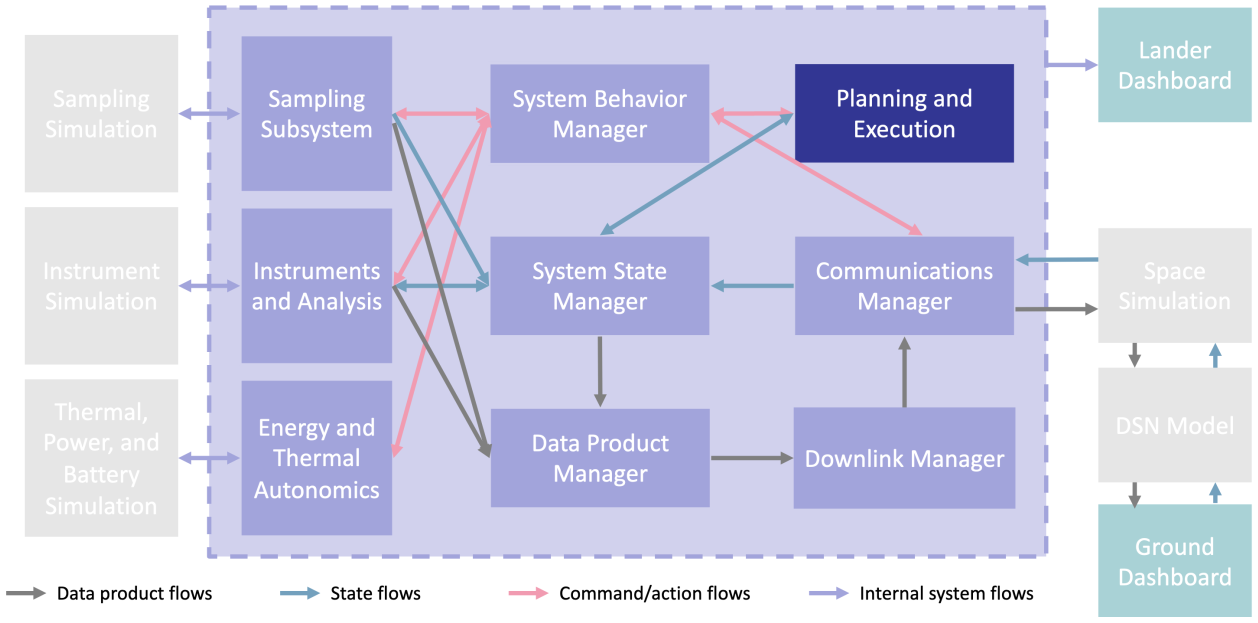


Fig. 9 Planning & Execution subsystem in the context of the ELAP Architecture

1. Lander & Environment Simulations

The simulation part of ELAP includes components representing the behavior of the sampling subsystem, onboard instruments, heating, battery and power management, and a communication subsystem. Each of these subsystems represents an interface between the lander itself and the outside environment, and as such includes components simulating the lander's state and behavior as well as components that model state and behavior of the environment that the lander is interacting with. The *Sampling Subsystem* is implemented by SAEL (Sampling Autonomy for Europa Lander), a functional autonomy software stack developed in parallel to ELAP to perform realistic simulation of robotic sampling behaviors, including estimations of the time and resource consumption of those behaviors. On the environment simulation side, SAEL uses the Dynamics Algorithms for Real-Time Simulation (DARTS), which included a lander model with mast, legs, a 5 DOF arm, plus tools, as well as a model of surface material properties [15].

[5]The prototype onboard software components include the SAEL sampling software, instrument analysis implementations, energy & thermal management, the communications subsystem, high-level behavior management, state management, and the planning & execution subsystem. In the case of this work, the planning and execution subsystem is implemented by MEXEC. It should be noted, however, that ELAP's modularity extends to supporting integration with other planning and execution platforms. For example, as part of the larger Europa Lander Autonomy Prototyping effort, the TRACE onboard executive [14] was also used extensively during development and evaluation.

The *Energy and Thermal Autonomics* subsystem models the state of the lander's batteries and simulates the power draw of all commanded and automatic lander activities. It also models the thermal state and simulates the heating of the lander's thermal zones, both directly (from explicitly commanded preheat and maintenance heat activities) and indirectly (via the waste heat generated from robotic activities). The environmental simulation for this subsystem consists of a thermal model of the Europan atmosphere, which exerts a significant cooling effect on the lander's components. The following instruments are simulated as part of ELAP: a gas chromatography mass spectrometer, microscope, and raman spectrometer to perform analysis on collected samples, as well as a seismometer and science camera to collect episodic data throughout the mission. Instrument simulation software is used to generate representative data at varying data volumes and science values, including biosignature levels (for sampling), quakes (for seismometry), or image changes (for science cameras), as well as content-based data reduction and summarization, to provide the system with choices of data to downlink that varied in both size and degree of scientific value.

The *Communications* subsystem includes models of the lander's radios, the light-time transmission delay between Earth and Europa, and the Deep Space Network, to provide realistic simulations of the transmission delay, bandwidth, and power draw for downlink and uplink activities. As work done by the lander only has value if the resulting data is communicated successfully back to Earth, communication scheduling was an important focus in the prototyping effort but falls outside the scope of this paper; instead, a complete description can be found in Appendix A.

C. Prototype Integration

A major part of our work on this project involved integrating MEXEC into Europa Lander Autonomy Prototype to attempt simulate full realistic surface missions. This integration and simulation effort shed light on a number of challenges that would be present in a real surface mission. One of the chief themes that we uncovered as part of this effort is that framing the scheduling of a hypothetical surface mission as a utility maximization problem requires that we have accurate estimates of the utilities of the tasks under consideration, as well as accurate estimates of the task parameter models (e.g. duration, energy use). In addition, in many realistic scenarios these values may be *context-dependent*. In this section, we describe our approach to these challenges and how they were implemented in the prototype.

1. Utility Estimation: Ranked Scores

Due to its short life expectancy, it is crucial that the Europa Lander minimize its time sitting idle. With long communication delays to Europa, the lander must often make the decision on what is best to do next with limited input from the ground team. Doing this requires an onboard estimation of the relative utility of the options presented to it. The utility of a task is defined by the utility of the data product it generates. The utility of a plan is a sum of the utility of the tasks in that plan. These utilities are used to evaluate what has been done so far, and what is planned to do next.

Specifically, we estimate the utility of data products, tasks, and plans using a ranked set of science scores. Each ranked score is a value from 0 to 1 and represents an independent measure of utility. The set of scores is ranked to allow some measures to dominate others when comparing utilities. For example, the ultimate goal is to downlink a data product that contains the results of an onboard science analysis used to detect a biosignatures (i.e. signs of life). The score for a data product with this analysis is given the highest rank. But there are other tasks that lead up to this analysis, each producing its own data product. The scores for these data products are included in the utility estimation, but are given lower ranks. Therefore, a data product or task with a higher score for the final analysis will always be considered to have higher utility, regardless of the other scores. Only when the scores are equal at one rank will the scores at a lower rank be compared.

Every utility measure, whether computed from actual data or predicted for future data, contains the same set of science scores. This set corresponds to the different types of data collected by the primary subtasks of the lander, and are ranked in the order in which these subtasks are performed. Specifically, the ranks from highest to lowest are:

1) Mission
2) Analysis
3) Post-collection
4) Pre-collection
5) Post-excavation
6) Pre-excavation
7) Default

The default rank is the lowest rank and allows some value to be given to a data product or task just for existing (when all other scores are 0). The cost of including the data or task will be considered separately. The pre- and post-excavation ranks are used for scores given to the data collected (e.g. images) before and after excavating a site. The pre- and post-collection ranks are used for scores given to the data collected before and after collecting a sample at a site. The analysis rank is used for scores computed from the final onboard analysis of the data. The mission score is given the highest rank, and is used to bias the planner towards performing certain actions according to the mission rules. For example, the mission score is used to enforce the rules of the subway map which ensure that no more than three samples are collected at a site and the first negative biosignature at a site directs the lander to collect samples at a different site. The basic idea is that when two data products have the same score for a step later in the process, the scores for the intermediate data leading up to that step are considered.

Once we have calculated (or predicted) the utility of a data product, this same utility can be assigned to the task that produces that data product. And when all tasks have been assigned a utility, we can estimate the utility of a partial plan by adding up the utility of all tasks in that partial plan. To do this, we add utilities by independently adding the scores at each rank. This maintains the dominance property, where a higher score at a higher rank is always considered better than any score at a lower rank. The resulting utility can then be used by the scheduler to compare or sort tasks and partial plans in a manner similar to lexicographical order. This ensures that the best possible tasks are considered, and that the search will consider higher utility plans before lower ones.

2. Utility Prediction: The Assessor

As part of the scheduling process, the MEXEC planner needs to predict utility values for hypothetical future science tasks. The predicted utility is different from an executed utility, which is a utility calculated from data products that have actually been generated. We specifically care about the utility value for the data that could be produced by a planned task, assuming that there is available time and resources to downlink that data. In addition, the utility of a planned task may be context-dependent, as it will depend on the history of prior observations. For example, if the lander has already excavated a site, we can calculate scores for the pre- and post-excavation imagery data that was collected. If multiple sites have been excavated, then each site will have these scores assigned. Because it plans into the future, MEXEC will consider collecting and analyzing hypothetical sample targets at each of these sites. As no information is known yet about these samples or the utility of imaging or analyzing them, until these tasks are executed, their utility will need to be predicted. To address this, we developed the *Assessor* module, a software component which takes as input a task and its context, and outputs a predicted utility value. The Assessor module evaluates the combined task and context against mission rules and constraints to compute an expected utility value for that task in that specific context. The context includes both existing data products relevant to the new task, as well as data products we would expect to receive from tasks that have been planned to support the new task. For example, when considering a new sampling task at a site that has already been excavated, the context includes the existing excavation data and any planned pre-collection imaging of the specific sampling target. The planner then sends the task and context combination to the Assessor, and uses the resulting utility to evaluate the search branch that adds this task.

3. Model Parameter Updates

Tasks in our system represent behaviors commanded by the planning and execution system, and model the values of impacts that we expect those behaviors to have on system states and resources. The high level of uncertainty inherent in a mission to Europa means that our system must be robust to inaccuracies in the prior estimates for these impact values. One way of ensuring such robustness is to generate planning and execution policies that can result in productive missions in spite of such inaccuracies[16]. We identified the following as potential sources of uncertainty in a Europa Lander mission:

1) Energy
    1) Tasks take more/less energy than expected
    2) Battery level drops unexpectedly (exogenous event)
    3) Battery level at mission end is unreliable

2) Excavation Depth
    1) Excavation rate slower than expected
    2) Unable to dig past threshold in one site
    3) Unable to dig past threshold in all sites
    4) Excavated trench collapses (i.e. reversed progress)
3) Heat
    1) System generates more/less heat than expected
    2) System cools more/less quickly than expected
    3) Sudden generation of heat/cooling (exogenous event)
4) Data & Communication
    1) Data rate higher/lower than expected
    2) Data buffers full
    3) Earth rise/set delayed (e.g. object blocking antenna)
5) Task Performance
    1) Any task fails
    2) Task succeeds, but does not have the expected impact (succeeds erroneously)
    3) Tasks take longer/shorter than expected

Additionally, as part of the Europa Lander Autonomy Prototype, we used feedback from the sampling subsystem to demonstrate in-situ updates to a subset of these values: task durations and energy impacts. The lander began its mission with uniform prior estimates for these task model parameters. After performing an initial sample collection task, the sampling subsystem provided actual measured values for the power draw and duration required for the activity. MEXEC used these values to update its modeled impact values for sampling tasks at the same site, by replacing them with the newly measured values.

D. Facilitating Ground Interactions

With substantial levels of onboard autonomy, we envision the need for a mission paradigm in which ground operators increasingly communicate *intent* (rules, constraints, and incentives that guide system behavior) rather than specific commands or sequences of commands. In this paradigm, the onboard system uses a *mission model* as its input, and the outcome is determined by the combination of model, system state, and execution state. Unlike with a plan or sequence, the model implies closed-loop execution with many possible outcomes. For this reason, we expect to need to develop new operation processes and tools with focus on building confidence that the model will result in outcomes that operators prefer. These tools would enable us to engage in *probabilistic reasoning* about the expected behavior of the onboard system, as we seek to maximize the likelihood of preferred outcomes and/or minimize the likelihood of undesirable outcomes across a wide variety of potential situations.

We envision an iterative process for authoring the onboard mission model: (1) generate the initial model, (2) use predictive tools to characterize expected outcomes (simulations/statistics), (3) use explanatory tools to understand causes of unexpected and/or undesirable outcomes, (4) Use advisory tools to make targeted updates to the model to result in more desirable outcomes, and (5) repeat until satisfied (or until time is up). This would require a modeling technology to facilitate the construction of plans with science intent and the iterative design process, as well as techniques for outcome/execution prediction, visualization, explanation, as well as advisory techniques (e.g., to fix undesirable behavior, add/change this constraint), to facilitate the operators learning process while helping them reassure that the spacecraft will complete the plan successfully. A similar process would be used for to both initial model generation and mid-mission model updates.

In the context of the ELAP prototype, we identified three classes of interactions through which the ground team could influence the lander's behavior at runtime: (1) changing onboard state values, (2) changing task parameters, and (3) altering the task network directly. Onboard states that would impact lander behavior included a flags indicating that a "ground hold" was present, preventing certain activities from being scheduled until it was lifted, as well as sampling-related information in the onboard database that could influence the ordering of activities (e.g. astrobiology signature results). Task parameters exposed to ground control include task model priors, such as the expected durations of robotic excavation and sampling tasks.

V. Empirical Evaluations

A. BSM-1 Simulation and Evaluation

To test our approach, we ran simulations of our planning and execution system commanding three variants of the BSM-1 reference mission described in Figure 1. The first is the base scenario in which each task consumes an amount of energy drawn from a normal distribution centered around its *a priori* expectation in the task network with a standard deviation of 10%. In the second variant, we bias this noise such that tasks are expected to consume 10% more energy than modeled. Finally, the third variant biases noise in the opposite direction, such that tasks are expected to consume 10% less energy. For each variant, we simulated each of the four planning/execution strategies discussed in our theoretical framework, and measured the utility achieved. In simulation, the failure probability of each task is uniform and independent. Each failure resolution mechanism is assumed to have a fixed cost and always succeed in resolving the issue. The data for each figure shows the mean utility achieved across 50 simulations of the scenario.

For our model calculations, we use Equations 1-5. We estimate our average utility per cost (u_{avg}) by analyzing plans generated by a prescient planner. This planner has perfect execution information *a priori*, so plan execution exactly matches the planner's predictions. Task failure probability is assumed to be $P(\text{fail}) = 0.1$, and we assume flexible execution is able to handle 30% of such failures for $P(\mathrm{FE}) = .03$, while replanning is able to handle an additional 60% of remaining failures for $P(\mathrm{replan}) = 0.042$ and ground can solve the rest for $P(\mathrm{ground}) = 0.028$.

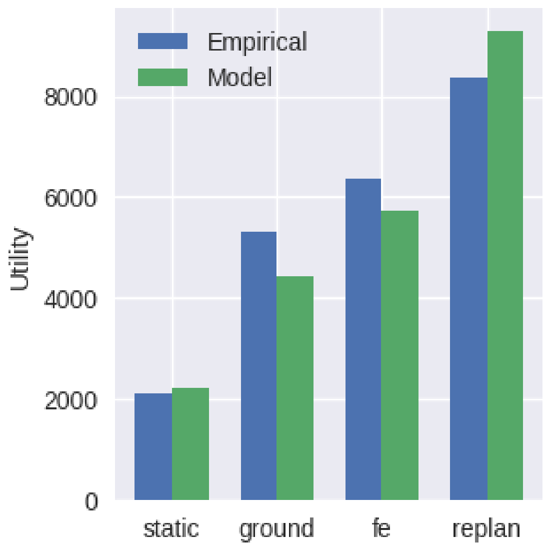


Fig. 10 Average utility achieved in simulation of the base Europa Lander domain for 4 planning strategies, compared to theoretical model predictions.

Our model predicts the `Static` strategy to perform poorly, since it has no failure resolution mechanisms and is thus likely to terminate quickly. By introducing a failure recovery mechanism, our model predicts the `Ground` strategy to improve performance considerably. However, this failure recovery mechanism is still fairly costly. The `FE` strategy introduces flexible execution to mitigate this. As such, our model predicts a higher utility achievement, since some set of failures are now resolved by a less costly mechanism. Finally, the `Replan` strategy is predicted to perform best of all the strategies. Like the `FE` strategy, it introduces another failure resolution mechanism and also introduces additional utility through plan optimization. When utility is discovered at execution time, the `Replan` strategy is able to exploit that discovery, where the other strategies are not.

In Figure 10, we compare the predictions of our model to the measured utility achievement of our system in simulation. We see that the four strategies qualitatively follow the model's predictions. The model uses u_{avg} as a way to estimate utility achievement based on power, smoothing performance across the entire execution into a linear model. However, in the Europa Lander domain, utility is primarily achieved only during communication events. Because the model views utility gain as purely linear, it is unable to capture the spikes in utility inherent in the domain.

In the Europa Lander domain, a site only needs to be excavated once, but multiple samples can be taken from a single excavation site. This means that the first sample taken at a site is much more costly than future samples. Because of this, if the system tends to run out of energy while attempting to sample a site for the first time, the model is likely to overestimate utility gain, since a significant portion of energy is used while no utility is gained. On the other hand, when the system can repeatedly sample from an existing site, the model underestimates utility. This behavior is prominently seen in the ground and FE strategies in Figure 10. Both strategies spend a significant portion of their execution repeatedly sampling from an excavation site, leading to higher utility gain than expected during these portions of the plan execution.

Another issue is the simplistic plan structure presumed by our model. In general, plan dependencies can be considered a set of graphs. For the Europa Lander domain, there are not merges (except for activities shared for efficiency such as downlinks), therefore plan structures look like forests or sets of trees. In our analysis model we presume that all activities in the plan form a single linear sequence. Because a plan is a set of trees, a failure in one tree would not stop execution in another tree - reducing the failure cost. Additionally, exogenous conditions such as Earth-in-view are required for downlink so that failure costs are non linear (a 1h delay could push a downlink to the next Earth-in-view 42 hours later or conversely a delay might not delay downlink at all as the downlink was waiting for a later Earth-in-view).

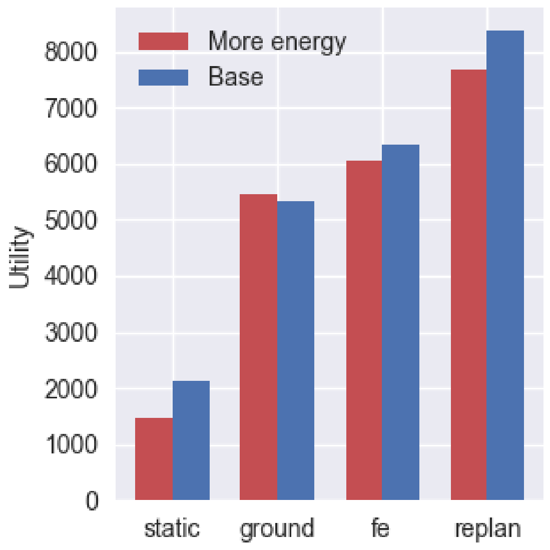
For the `Replan` strategy, we also consider the effects of utility discovery and plan optimization in replanning. To determine a value for d, the number of times that utility discovery can be exploited, we calculate an upper bound for this value based on the total energy available to the system. However, the system may not be able to take advantage of utility discovery this number of times, since it may run into too many task failures, or the planner may simply choose to complete other tasks. Thus, the calculations for our model tend to overestimate the value of utility discovery in the replanning strategy.

Next, we consider the effects of biased noise on the utility achieved by our system. First, we examine the scenario where all tasks use 10 percent more energy on average than expected. A comparison of this scenario and the base scenario is shown in Figure 11a. Naively, we might expect utility in each scenario to decrease by about 10 percent. However, because utility is achieved in spikes through the completion of fairly lengthy chains of tasks, events have an impact on utility only if they increase or decrease the probability of successfully completing a chain of tasks. In the "more energy" scenario, the ground strategy is largely unaffected.

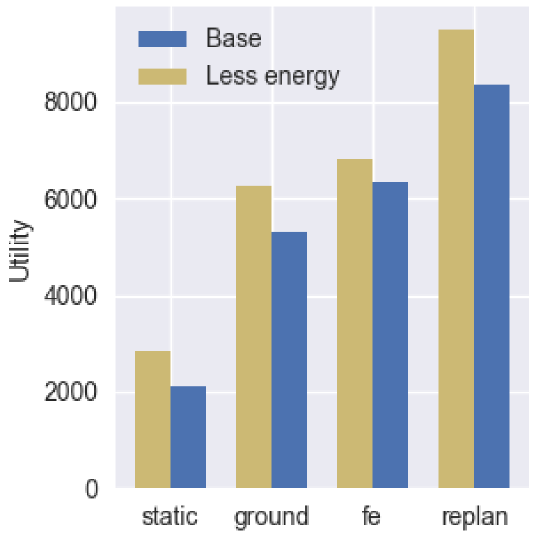
The `Replan` strategy is affected more heavily, since a lower pool of energy available limits the strategy's ability to take advantage of discovered utility. On the other hand, because it is able to replan, it can make use of lower cost actions such as Seismograph/Panorama tasks to gain utility despite lacking the energy to complete a sample.

Finally, we consider the scenario where tasks take 10 percent less energy than expected (Figure 11b). Here, the ground strategy improves considerably in performance, while FE improves at a lower clip. This is consistent with what we see in the previous scenario. The `Ground` strategy is able to benefit significantly from the extra energy and complete an extra sample cycle, while FE is not as close to this boundary and thus is not affected as strongly.

The `Replan` strategy also sees significant benefits from extra energy. Extra energy enables additional samples, whose benefit is amplified by the potential for utility discovery. In addition, the `Replan` strategy is able to integrate knowledge of the additional energy during execution time as it updates state predictions with the reality on the ground. Thus, instead of settling for a Seismograph/Panorama task, as might occur in the base or high energy use scenarios, the



(a) Average utility achieved in simulation of the Europa Lander domain where all tasks take 10% more energy than expected, compared to empirical results in the base domain.



(b) Average utility achieved in simulation of the Europa Lander domain where all tasks take 10% less energy than expected, compared to empirical results in the base domain.

Fig. 11 Results from the BSM-1 experiment

replan strategy is more often able to process a sample.

B. ELAP Simulation and Evaluation

1. Simulated Scenarios

We designed and executed a set of scenarios intended to represent variations in conditions and behavior that a Europa Lander mission might plausibly encounter. These scenarios were selected as being representative of important classes of variation in mission conditions and lander performance that the system must be capable of responding to in a robust manner. They cover such areas as: maximizing utility of data returned to Earth under a wide variety of sampling outcomes, incorporating state feedback during execution, and changing task model parameters (both impacts and constraints) during execution. These scenarios, and many others, are being studied to better understand the challenges in deploying an operational onboard autonomy system for the Europa Lander Mission Concept. In each scenario, the onboard autonomy would be expected to carry out the mission to completion with as little loss to over-all achieved utility as possible:

1) *All samples positive*: This scenario is the “baseline” against which performance on all other scenarios could be compared. In this case, all aspects of the lander’s state and resource use closely match their modeled values and all samples produce positive biosignature results.
2) *All samples negative*: Similar to the baseline case, the initial state and lander performance match expectations, but in this case all samples produce negative biosignature results.

3) *Low starting battery state-of-charge*: In this case, the state-of-charge at the start of the mission is 50
4) *Ground input changes excavation order*: In this case, ground operators update the relative science values of the excavation sites after they have been assigned an initial science value via onboard algorithms.
5) *First sample negative*: In this case, the first sample results in a negative biosignature; all following samples are positive.
6) *Ground input changes mission model*: In this scenario, ground operators uplink a new onboard planning model that specifies that the lander should sample from a single target at each site regardless of science value.
7) *Increased decisional data volume*: In this case, the seismometer, science camera, and sampling instrument analysis data product generation are all updated to produce significantly more decisional data.
8) *Sudden low battery*: In this scenario, the available state-of-charge is reduced to 10
9) *Higher comm energy allocation*: In this case, limits on per-sol energy that can be spent on communication drastically increased (35x) from baseline, representing a case where there are essentially no mission-level constraints on comms energy allocation driving the relative effort spent on communication vs science activities.
10) *Slower heating*: In this case, the simulated ambient temperature is dropped to 85K (from 100K at baseline), resulting in additional time and resources needing to be spent on heating.
11) *Sudden downlink slowdown*: In this case, the downlink rate is reduced from 50kbps to 5kbps after the third sample is collected, simulating a scenario in which the lander is instructed by ground operators to communicate at a slower rate due to poor Earth/DSN conditions.
12) *Increased mandatory data volume*: In this case, the seismometer, science camera, and sampling instrument analysis data product generation are all updated to produce significantly more raw mandatory data.

2. Findings / Lessons Learned

Our results for the prototype integration and scenario demonstrations confirm that a system-level autonomy approach like the one we developed here is capable of successfully commanding a complex surface mission. The MEXEC-based Planning & Execution component was able to dispatch commands for sampling, analysis, periodic imaging, ongoing seismometry, and communications in a dynamic way while managing onboard resources such as time, energy, and data volume to ensure a measure of productivity under a wide range of execution circumstances, successfully handling each of the scenarios listed above. Overall, the planner was able to meet a majority of the scenario success criteria, and successfully achieved the system design objectives. This effort also highlighted the value of having the ability to perform high-fidelity mission simulations with a system like ELAP. The scenario execution and evaluation process provided a concrete way of evaluating our approach to autonomy and its implementation, as well as our hypothetical mission design, and the interaction between the two.

This experimentation process revealed a number of cases in which the implementation of our Planning & Execution module diverged from our design intent, including in the modeling of task utilities and in the constraints on what data can be downlinked at various stages of the mission; future efforts should first focus on repairing these inconsistencies. The simulations then become a valuable tool to evaluate performance across multiple dimensions: mission design, degree and nature of autonomy, degree and nature of ground interaction.

An approach to onboard autonomy that centers around utility-maximizing search for scheduling tasks, like the one we developed for this project, requires new ways of thinking about mission design and operations. Teams that are used to specifying agent behavior in a rule-based manner or via flowchartsstate diagrams will need to be made aware of the new paradigm for influencing the autonomous agent's behavior (using utility values, task and resource constraints), the potential drawbacks (less visibility into the decision-making process, less influence on specific behaviors), and the potential benefits (ultimately: higher over-all science return). Autonomy developers will need to work to increase users' confidence in such systems through proof and demonstration, and will also need to work to provide tools to support the types of control that mission designers and ground operators find most valuable as well as tools for explainability to help operators and scientists understand the agent's decisions during and after mission operations.

VI. Future Work

The above approaches to uncertainty are reactive in that they adjust execution (or reschedule) in response to execution variations. An alternative approach is to proactively plan/or execute in ways that are known to be resilient to environmental and execution variation. The approach outlined by Basich and others [17] utilizes this approach. In this approach, plans are explicitly evaluated against against resource varying contingencies (specifically variations in available energy from predicted levels) and search bias it towards plans that perform best across the possible outcomes of more, less, or predicted energy consumption/availability. One would expect such approaches to have the potential to outperform reactive approaches to execution variation.

It is also worth noting that validation and verification of an autonomous systems is a tremendous challenge. To date spaceflight of significant autonomy [18] has involved extensive validation and verification efforts relying on the full suite of formal methods, informal methods, and testing.

VII. Related Work

The work proposed in this paper comprises only a part of the effort made for the overall Europa Lander mission concept. Other efforts include the autonomous sampling effort [15], the Blackbird simulator [19], the ground operation designs, and the larger autonomy prototyping effort, for details on which we direct the reader to Wagner et al. [13]. The work proposed here focuses on the system level autonomy aspect for the proposed Europa Lander pre-project effort, in particular the planning and execution framework.

There have been very few flights of system level autonomy on space missions. This system level autonomy is typically implemented as a planner/scheduler and executive or in some cases as an executive alone. The scarcity of flights is because system level autonomy implies control over most if not all mission activities and therefore requires a very high degree of confidence in the autonomy/scheduling system (see prior note on the importance of validation and verification of space autonomous systems [18]).

The Remote Agent Experiment flew an onboard planner/scheduler as an experiment for two periods totalling approximately 48 hours in 1999 onboard the Deep Space One spacecraft [20]. The Remote Agent used a batch planner to generate a temporally flexible plan that was used by a reactive executive controller [21] to provide robust plan execution but did not consider utility in plan generation and did not perform continuous replanning due to the computational expense and long planning time. The CASPER planner flew as the primary operations system onboard the Earth Observing One (EO-1) mission for over a dozen years from 2004-2017 [22] successfully executing tens of thousands of observations and dramatically enhancing the EO-1 science mission. This onboard planner was also integrated with other data feeds (ground and space) as part of multiple sensorwebs most notably to track flooding [23] and volcanic activity worldwide [24]. However, the EO-1 mission was significantly less complex than the proposed Europa Lander mission and also as an orbiter did not involve significant physical interaction with the environment. The Intelligent Payload EXperiment (IPEX) Technology demonstration cubesat also flew the CASPER planner for over 1 year [25]. More recently the Mexec planner flew in a limited capacity onboard the ASTERIA Cubesat [4]. However all of these were relatively simple spacecraft and/or limited scope and duration. Both the Spitzer Infra red telescope Facility (SIRTF) and the James Webb Space Telescope (JWST) [26] operationally utilize a form of flexible execution which handles timing variations (such as in acquisition of guide stars) and therefore improve efficiency of operations.

A notable exception to the above is the M2020 onboard planner [27]. The M2020 onboard planner targets a surface robotic mission which has a much more intimate interaction with their environment and thermal and energy management considerations for the Perseverance rover are a major part of rover operations and therefore required special treatment for the onboard scheduler. As such this planner must handle similar robotic considerations as for the Europa Lander Mission Concept.

VIII. Conclusions

In this paper, we have presented a utility-based hierarchical task network planning approach to autonomy targeted at a Europa Lander mission concept. In this approach, the system uses flexible execution and utility optimization-based planning to respond to robotic execution variations and variable science outcomes. We empirically validated that the proposed approach outperforms standard baseline approaches used in space domains. Additionally, we have developed an analytical model to characterize the benefits of autonomous response for the general case and studied this in the Europa Lander Mission Concept. We tested the proposed system in a series of extensive software simulations as part of the larger Europa Lander Mission Concept Autonomy Prototyping effort. These tests confirm that such an autonomous system appears feasible and would significantly enhance a potential Europa Lander mission.

Acknowledgments

We would like to thank the members of the entire Europa Lander pre-project at JPL, and especially the Europa Lander Autonomy Prototype team; the software implementation and testing described in this work would not have been possible without their support and expertise. In no particular order, we would like to acknowledge Dustin Aguilar, Yuliya Marchetti, Scott Tepsuporn, Grace Lim, Todd Litwin, Peter di Pasquale, Igor Uchenik, Jeff Pamer, Joseph Bowkett, Brett Kennedy, Garrett Johnson, Paul Backes, Hrand Aghazarian, Anna Sabel, Vandi Verma, Jacob Kosberg, William Seto, David Kim, Dan Delany, Jeffrey Edlund, Yumi Iwashita, Jenette Sellin, Masahiro Ono, and JP De la Croix The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Hand, K. P., *Report of the Europa Lander science definition team*, National Aeronautics and Space Administration, 2017.

[2] Agrawal, J., Chi, W., Chien, S. A., Rabideau, G., Kuhn, S., Gaines, D., Vaquero, T., and Bhaskaran, S., "Enabling Limited Resource-Bounded Disjunction in Scheduling," *Journal of Aerospace Information Systems*, Vol. 18:6, 2021, pp. 322–332. URL `https://doi.org/10.2514/1.I010908`.

[3] Gaines, D., and et al, "Productivity challenges for Mars rover operations," *Proceedings of 4th Workshop on Planning and Robotics (PlanRob)*, London, UK, 2016, pp. 115–125.

[4] Troesch, M., Mirza, F., Hughes, K., Rothstein-Dowden, A., Bocchino, R., Donner, A., Feather, M., Smith, B., Fesq, L., Barker, B., and Campuzano, B., "MEXEC: An Onboard Integrated Planning and Execution Approach for Spacecraft Commanding," *Workshop on Integrated Execution (IntEx) / Goal Reasoning (GR), International Conference on Automated Planning and Scheduling (ICAPS IntEx/GP 2020)*, 2020. URL `https://ai.jpl.nasa.gov/public/papers/IntEx-2020-MEXEC.pdf`, also presented at International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2020) and appears as an abstract.

[5] Nau, D. S., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F., "SHOP2: An HTN Planning System," *Journal of Artificial Intelligence Research*, Vol. 20, 2003, p. 379–404. https://doi.org/10.1613/jair.1141, URL http://dx.doi.org/10.1613/jair.1141.

[6] Goldman, R. P., and Kuter, U., "Hierarchical Task Network Planning in Common Lisp: the case of SHOP3," *Proc 12th European Lisp Symposium*, Zenodo, 2019, pp. 73– 80. https://doi.org/10.5281/zenodo.2633324, URL https://doi.org/10.5281/zenodo.2633324.

[7] Verma, V., Gaines, D., Rabideau, G., Schaffer, S., and Joshi, R., "Autonomous Science Restart for the Planned Europa Mission

with Lightweight Planning and Execution," *International Workshop on Planning and Scheduling for Space (IWPSS 2017)*, Pittsburgh, PA, 2017, pp. 193–201. URL https://ai.jpl.nasa.gov/public/papers/verma_iwpss2017_autonomous.pdf.

[8] Chi, W., Chien, S., Agrawal, J., Rabideau, G., Benowitz, E., Gaines, D., Fosse, E., Kuhn, S., and Biehl, J., "Embedding a Scheduler in Execution for a Planetary Rover," *International Conference on Automated Planning and Scheduling (ICAPS 2018)*, Delft, Netherlands, 2018. URL https://ai.jpl.nasa.gov/public/papers/chi_icaps2018_embedding.pdf, also appears at International Symposium on Artificial Intelligence, Robotics, and Automation for Space (ISAIRAS 2018) as an abstract.

[9] Rabideau, G., and Benowitz, E., "Prototyping an Onboard Scheduler for the Mars 2020 Rover," *Intl Workshop on Planning and Scheduling for Space (IWPSS 2017)*, Pittsburgh, PA, 2017. URL https://ai.jpl.nasa.gov/public/papers/rabideau_iwpss2017_prototyping.pdf.

[10] Agrawal, J., Chi, W., Chien, S. A., Rabideau, G., Gaines, D., and Kuhn, S., "Analyzing the Effectiveness of Rescheduling and Flexible Execution Methods to Address Uncertainty in Execution Duration for a Planetary Rover," *Robotics and Autonomous Systems*, Vol. 140 (2021) 103758, 2021. URL https://doi.org/10.1016/j.robot.2021.103758.

[11] Chien, S., Johnston, M., Policella, N., Frank, J., Lenzen, C., Giuliano, M., and Kavelaars, A., "A generalized timeline representation, services, and interface for automating space mission operations," *International Conference On Space Operations (SpaceOps 2012)*, Stockholm, Sweden, 2012. URL https://ai.jpl.nasa.gov/public/papers/chien-spaceops2012-generalized.pdf.

[12] Chien, S. A., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling." *International Conference on AI Planning and Scheduling (AIPS)*, 2000, pp. 300–307.

[13] Wagner, C., Mauceri, C., Twu, P., Marchetti, Y., Russino, J., Aguilar, D., Rabideau, G., Tepsuporn, S., Chien, S., and Reeves, G., "Demonstrating Autonomy for Complex Space Missions: A Europa Lander Mission Autonomy Prototype," *Journal of Aerospace Information Systems*, 2023, pp. 1–21.

[14] de la Croix, J.-P., and Lim, G., "Event-Driven Modeling and Execution of Robotic Activities and Contingencies in the Europa Lander Mission Concept Using BPMN," 2020.

[15] Bowkett, J., Nash, J., Kim, D. I., Kim, S.-K., Thakker, R., Brinkman, A., Cheng, Y., Willson, R., Lim, C., Gaut, A., Jain, A., Gildner, M., Emanuel, B., and Backes, P., "Functional Autonomy Challenges in Sampling for an Europa Lander Mission," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–8. https://doi.org/10.1109/AERO50100.2021.9438298.

[16] Basich, C., Russino, J., Chien, S., and Zilberstein, S., "A Sampling Based Approach to Robust Planning for a Planetary Lander," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, Kyoto, Japan, 2022. URL https://ai.jpl.nasa.gov/public/documents/papers/basich-iros2022.pdf.

[17] Basich, C., Wang, D., Chien, S., and Zilberstein, S., "A Sampling-Based Optimization Approach to Handling Environmental Uncertainty for a Planetary Lander," *International Workshop on Planning and Scheduling for Space (IWPSS)*, 2021. URL https://ai.jpl.nasa.gov/public/papers/Basich_IWPSS2021_paper_3.pdf.

[18] Chien, S., "Formal Methods for Trusted Space Autonomy, Boon or Bane?" *NASA Formal Methods Symposium*, 2022. URL https://ai.jpl.nasa.gov/public/documents/papers/chien-nfm-2022.pdf.

[19] Kim, S. Y., Roffo, K., Ye, S. C., Tan-Wang, G., Laubach, S., Pyrzak, G., and Reeves, G., "Designing for operating autonomous space missions: Concept of operations design for europa lander using mission-level modeling and simulation," *ASCEND 2021*, 2021, p. 4116.

[20] Jónsson, A. K., Morris, P. H., Muscettola, N., Rajan, K., and Smith, B. D., "Planning in Interplanetary Space: Theory and Practice." *AIPS*, 2000, pp. 177–186.

[21] Pell, B., Gat, E., Keesing, R., Muscettola, N., and Smith, B., "Robust periodic planning and execution for autonomous spacecraft," *IJCAI*, 1997, pp. 1234–1239.

[22] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davis, A., Mandl, D., Frye, S., Trout, B., et al., "Using autonomy flight software to improve science return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 4, 2005, pp. 196–216.

[23] Chien, S., Mclaren, D., Doubleday, J., Tran, D., Tanpipat, V., and Chitradon, R., "Using Taskable Remote Sensing in a Sensor Web for Thailand Flood Monitoring," *Journal of Aerospace Information Systems (JAIS)*, Vol. 16, No. 3, 2019, pp. 107–119. https://doi.org/10.2514/1.I010672, URL https://doi.org/10.2514/1.I010672.

[24] Chien, S. A., Davies, A. G., Doubleday, J., Tran, D. Q., Mclaren, D., Chi, W., and Maillard, A., "Automated Volcano Monitoring Using Multiple Space and Ground Sensors," *Journal of Aerospace Information Systems (JAIS)*, Vol. 17:4, 2020, pp. 214–228. URL https://doi.org/10.2514/1.I010798.

[25] Chien, S., Doubleday, J., Thompson, D. R., Wagstaff, K. L., Bellardo, J., Francis, C., Baumgarten, E., Williams, A., Yee, E., Stanton, E., et al., "Onboard autonomy on the intelligent payload experiment cubesat mission," *Journal of Aerospace Information Systems*, Vol. 14, No. 6, 2017, pp. 307–315.

[26] Zonak, S., "Event-driven operations, OSS and JWST," *STSCI Newsletters*, 2020. URL https://www.stsci.edu/contents/newsletters/2020-volume-37-issue-02/event-driven-operations-oss-and-jwst.

[27] Gaines, D., Chien, S., Rabideau, G., Kuhn, S., Wong, V., Yelamanchili, A., Towey, S., Agrawal, J., Chi, W., Connell, A., Davis, E., and Lohr, C., "Onboard Planning for the Mars 2020 Perseverance Rover," *16th Symposium on Advanced Space Technologies in Robotics and Automation*, 2022. URL https://ai.jpl.nasa.gov/public/documents/papers/M2020-OBP-ASTRA-2022-Final.pdf.

A. Communication Scheduling: The Downlink Manager

Recall that ultimately the work done by the lander only has value if the resulting data is communicated successfully back to Earth. In other words, a data-generating task should only performed if the resulting data can be downlinked before the end of the mission. Otherwise, time and resources will be wasted performing the task with no benefit. Therefore, each time a new task is considered to be included in the plan, the resulting partial plan must be evaluated to determine if there will be enough time and resources not only to perform the task, but to downlink all data products including any from the new task. Only then is the new partial plan included as a candidate for further expansion by the planner.

To solve the sub-problem of downlink allocation as a service to planning, we developed an independent software component referred to as the *Downlink Manager*. In addition to determining the feasibility of downlinking a set of data products, the Downlink Manager also assigns data products to downlink opportunities according to a set of constraints and preferences. Downlink opportunities occur at fixed communication windows, which at a minimum include time for uplink to the lander. The assignment of a data product to a downlink opportunity is limited by the following set of constraints and properties:

- Earth in-view: opportunities for communication only exist when the lander is in view of a ground station on Earth
- Energy allocation: each sol is assigned an energy allocation for performing downlink tasks, as a way of strategically controlling energy used by downlinks
- Thermal limits: communication hardware heats up when used and cannot be used over a specified temperature
- Downlink rate: the duration and capacity of the downlink depends on the downlink rate
- Data product size: the number of bytes that need to be downlinked for the data product
- Data product creation time: the downlink opportunity must exist after the data product is created
- Data product downlink priority: the urgency for downlinking the data product (downlink now, ASAP, or before mission end)

In addition, a set of preferences dictate which assignments are better than others. Specifically, the following preferences were defined for downlinking data products from the lander:

- Science utility: prefer to downlink data products with higher utility values assigned or predicted by the Assessor
- Earliest downlink: all else being equal, prefer to downlink at the earliest possible opportunity

Note that, because re-planning is occurring continuously throughout the mission, executed tasks and existing data products must be considered along with any candidate new tasks and data products. This involves knowing the actual properties of existing data products (e.g. size, utility) as well as predicting the same properties for those that we expect to generate. These properties are used to predict the duration and energy costs for hypothetical future downlink tasks. As with science task utility, downlink task duration and energy costs are context-dependent; in particular, they depend on the size and type of onboard data that are expected to be available for transmission at the start time of the proposed downlink. The Downlink Manager takes as input a set of downlink opportunities and a set of data products, and outputs a list of downlinks with the individual data products assigned to them. When the planner considers a new task, the Downlink Manager is used to solve the new problem. If the data product generated by the new task is not assigned to a downlink, then that task is not scheduled (i.e. that branch is pruned from plan search). By rejecting candidate tasks whose data products cannot be downlinked, we ensure that the spacecraft will cease data-generating activities with enough energy remaining to communicate all required data to the ground.

The problem addressed by the Downlink Manager is largely a packing problem: data products of different sizes need to be packed into downlinks with difference capacities. Many of the data products can be "packed" into any downlink opportunity "bin", where each bin size is defined by the maximum downlink capacity imposed by the constraints. Thus, maximizing the fit of items into bins is the primary objective. Specifically, we transform our problem into the Multiple Knapsack Problem, and solve it using Constraint Integer Programming (CIP) using a library from Google's OR Tools. The library's open source license, use of the C programming language (preferred for flight systems), and large user base made it an ideal choice for our prototype implementation. The CIP problem can be stated as follows:

Given a set of values (V) and sizes (S) for data products, a set of capacities (C) for downlinks, and a set of unavailability indicators (U) for data product and downlink pairs:

$$V_i \in \mathbb{Z}, i = 1, ..., k \tag{6}$$

$$S_i \in \mathbb{Z}, i = 1, ..., k \tag{7}$$

$$C_j \in \mathbb{Z}, j = 1, ..., l \tag{8}$$

$$U_{ij} \in \{0, 1\}, i = 1, ..., k, j = 1, ..., l \tag{9}$$

Compute downlink assignments (X):

$$X_{ij} \in \{0, 1\}, i = 1, ..., k, j = 1, ..., l \tag{10}$$

That maximize the values of assigned data products:

$$\sum_{i=1}^{k} \sum_{j=1}^{l} V_i X_{ij} \tag{11}$$

Subject to the constraints:

$$\sum_{j=1}^{l} X_{ij} \leq 1, i = 1, ..., k \tag{12}$$

$$\sum_{j=1}^{l} U_{ij} X_{ij} = 0, i = 1, ..., k \tag{13}$$

$$\sum_{j=1}^{l} S_i X_{ij} \leq C_j, i = 1, ..., k \tag{14}$$

The input variable V_i is the integer value of data product i assigned based on the preferences mentioned earlier (science utility and downlink earlier time). The input variable S_i is the integer size (in bytes) of data product i. The input variable C_j is the capacity (in bytes) of downlink opportunity j. And the input variable U_{ij} is either 0 or 1 where 1 indicates the unavailability to downlink data product i during opportunity j (e.g. because the data product will not be created by the time of the downlink).

The output variable X_{ij} is a 0 or 1 representing whether or not data product i was assigned to downlink j. The objective function is simply to maximize the sum of the value of each data product that was assigned. The first constraint states that each data product can be assigned to at most one downlink. The second constraint ensures that each data product, if assigned, was not assigned to a downlink that is unavailable for that data product. Availability is determined based on the creation time of the data product and the time of the downlink opportunity. The third constraint prevents data products from overflowing the capacity of the downlink opportunity. Capacities are assigned based on the constraints mentioned earlier (e.g. energy allocation). The constraint imposed by downlink priority is handled as a special case. High-priority data products, those indicating the need to downlink immediately or in the next opportunity, do not have a choice for downlink assignment, so the CIP solver was only used to assign the lower-priority data products.

With a new set of assignments of data products to downlinks, the planner can continue the evaluation of the candidate task. If the data product for the task was not assigned to a downlink, the specific partial plan under consideration is rejected. Otherwise, the downlinks in the partial plan are updated to reflect new predictions for durations and energy use. This updated partial plan is then included in the list of partial plans that will be considered for further expansion.