Analyzing the Efficacy of Flexible Execution, Replanning, and Plan Optimization for a Planetary Lander

Daniel Wang, Joseph A. Russino, Connor Basich, and Steve Chien

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109

Abstract

Plan execution in unknown environments poses a number of challenges: uncertainty in domain modeling, stochasticity at execution time, and the presence of exogenous events. These challenges motivate an integrated approach to planning and execution that is able to respond intelligently to variation. We examine this problem in the context of the Europa Lander mission concept, and propose a planning and execution framework that responds to feedback and task failure using two techniques: flexible execution and replanning with plan optimization. We develop a theoretical framework to predict the value of each of these techniques, and we compare these predictions to empirical results generated in simulation. We demonstrate that an integrated approach to planning and execution that is grounded in flexible execution, replanning, and utility maximization will be an enabling technology for future tightly-constrained planetary surface missions.

Introduction

When integrating AI planning into robotic applications, planners are consistently challenged by variation in execution and uncertainty in the quality of our environment models. In space-based applications, this is especially challenging because the environment is largely unknown, reducing the quality of our a priori models of the world. To address these problems, we describe an integrated approach to planning and execution in an unknown, unpredictable environment. First, we define a theoretical framework to examine the value of two integrated planning and execution techniques: flexible execution and replanning with plan optimization. We discuss this framework in the context of the Europa Lander mission concept. Finally, we compare the predictions of the model to empirical results in a Europa-like simulation environment.

The primary empirical context of our model is a mission concept to perform *in situ* analysis of samples from the surface of the Jovian moon Europa (Hand 2017). Unlike prior NASA missions, a priori domain knowledge is severely limited and uncertain, and communication with Earth is limited by long blackout periods (about 42 hours out of every 84 hours). Consequently, a successful mission requires a planning and execution framework that is highly efficient[1] , robust to unprecedented levels of uncertainty, and still capable of maximizing its overall utility. On the other hand, the Europa Lander concept has a fairly rigid definition of what actions the lander must perform in order to produce utility. Our planning algorithm leverages this domain-specific knowledge by making use of a hierarchical task network (HTN) and using heuristic-guided search to examine various task combinations to maximize utility. The ultimate goal for a Europa Lander would be to analyze surface material and communicate the resulting data products back to Earth. To reward accomplishment of these goals, we assign utility to tasks such as sample excavation and seismographic data collection, but do not receive this utility until the lander communicates the data down to Earth. In the HTN framework, this means that tasks in a hierarchy produce utility only if the full hierarchy is executed.

For our empirical evaluation, we base our planning system on MEXEC, an integrated planner and executive first built for NASA's Europa Clipper mission (Verma et al. 2017). We compare four approaches to planning on the Europa Lander problem similar to those used in prior missions: a static plan without failure recovery mechanisms, a static plan with ground input for failure recovery (Gaines et al. 2016), flexible execution without replanning, and flexible execution with replanning (Rabideau and Benowitz 2017). We explore the value of flexible execution and replanning with plan optimization, and examine these techniques' effects on utility in these scenarios. We demonstrate that, true to our model's prediction, each technique shows significant improvement in utility achievement in the Europa Lander domain.

Domain Description

The primary goal of the Europa Lander mission concept is to excavate and sample the surface, analyze the sampled material for signs of biosignatures, and communicate that

[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 (Agrawal et al. 2019) is only allocated a portion of the computing cycles onboard the RAD750 resulting computation *several thousand times* slower than a typical laptop.

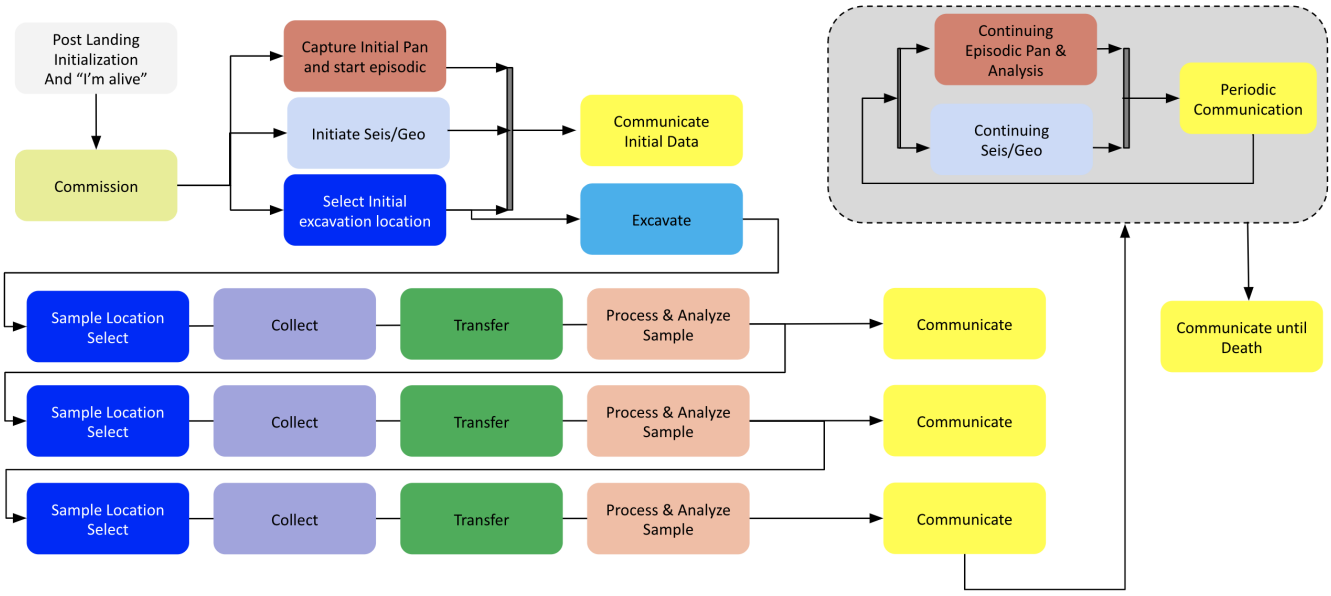

Figure 1: A task network for the Europa Lander mission concept. The diagram represents a potential execution trace of the mission that would fulfill baseline requirements.

data back to Earth (Hand 2017). 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 concept mission clearly defines the sequence of actions required to achieve these goals. Figure 1 displays the strong dependency structure inherent to the Europa Lander concept mission. In order to sample, the lander needs to have excavated a trench; in order to analyze, the lander needs to have collected a sample; etc.

As a minimum requirement, the lander should excavate a trench in the Europan surface, collect three samples from that site, analyze those samples, and return that data to Earth. The basic requirements of a mission would require only a single site to be excavated. However, there is value in excavating additional sites, because the material at different sites may possess different properties. On the other hand, the lander may choose to resample the same location, for example, in order to verify the discovery of a biosignature. In the baseline mission concept, all three of the lander's samples are chosen from the same target. Note that after the first site is excavated, no further excavations are needed to sample from that trench; all three sampling activities can share a single excavation site. After excavation and sample collection, samples must be transferred into scientific instruments that analyze the material and produce data products. Then, for a mission to achieve any actual utility, those data products must be 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 lesser 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 utility is only achieved when data is downlinked back to Earth. 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 plan accordingly to maximize utility in face of 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.

Approach

We design our planning system to respond intelligently to stochasticity at execution time, since we expect this to be a significant factor in our domain. Planning and execution are integrated in our approach, in order to respond to variation

and therefore better optimize overall utility achieved. We achieve this integration through the use of two techniques: flexible execution and replanning with plan optimization.

Flexible Execution

Flexible execution is a lightweight rescheduling algorithm that runs at a much higher cadence than the planner. This algorithm has two main properties: (1) it is significantly less costly than replanning, and (2) it is significantly less powerful than replanning. Despite its limited scope, flexible execution is valuable because it can be run so frequently. This 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 (Chi et al. 2018). 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 failing the task, we allow it to wait for some amount of time for this inconsistency to resolve. Such a situation might occur, for example, if previous dependencies are unexpectedly delayed. We then push the start time of the task forward in the plan. Task push is implemented as a callback that is run before a task is dispatched to the execution system. 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*. After a task completes with a failure code, flexible execution can immediately re-schedule the task if its preconditions are still met. The plan is then updated to account for the new predicted end time of the task, as well as its additional resource usage. Here, the system short-circuits a simple failure response, avoiding planning costs for failures whenever possible.

In the context of the Europa Lander domain, flexible execution offers significant value despite its simplicity. Because significant noise is expected in resource impacts, providing a low-cost method of handling mismatches in resource use predictions often avoids costs associated with either replanning or waiting for ground input. For example, if heating a joint on the lander is slower than predicted, flexible execution may handle this by re-triggering the heating operation or delaying arm movement, either of which would be sufficient to resolve the issue.

Replanning with Plan Optimization

For more complex failure responses, simple retries may be insufficient. In these cases, we turn to replanning during execution. Replanning allows the system to make use of online state updates, responding to variation from the original plan's predictions. Our framework measures the value of each resource being modeled, and assigns that value to the given resource in the planning model. Then, when replanning, the planner uses the actual, measured value of the state, rather than the previous predicted value. This allows the system to update its goals according to what is realistically possible given the current state measurements of the system. When tasks fail, their predicted state impacts are usually not realized. Replanning provides a mechanism to respond to these problems in a more complex manner than retrying the task. For example, excavation of the Europan surface is a complicated task with many modes of failure. Retries or delays may be insufficient responses to these failure modes, which may require additional actions to be taken.

In addition to this, replanning allows the system to make use of additional knowledge gained at execution time. This may take the form of task model updates and utility adjustments. For example, during execution time, the lander may discover that a task consumes more energy than expected, or that it produces more valuable data than expected. In the Europa Lander domain, the system might discover a biosignature at a sampling location, which would drastically change the site's utility. This is where plan optimization comes into play. By updating the task models, replanning can take execution-time knowledge into account and generate plans that produce more utility. Replanning thus improves overall utility achievement through two mechanisms: more advanced failure recovery, and plan optimization given execution-time knowledge.

Theoretical Framework

We define our planning problem as follows. We provide our planner with a set of tasks $T = \{t_0, t_1, ..., t_n\}$. Each task is represented by a tuple $t_k = \{c_k, u_k, d_k, P, I\}$ 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 preconditions. These may be based on resource values, or on the execution state of dependency tasks.
- I is the set of its impacts on resource timelines.

This matches the timeline representation of execution state used by (Verma et al. 2017). 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 wish 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.

In our framework, we examine four planning and execution strategies: static, ground, FE, and replan. Using the static strategy, a plan is generated before execution time, then executed without change. No failure responses are enabled, so any task failure results in the termination of plan execution. In the ground strategy, we introduce a mechanism for failure resolution: waiting for ground input. We assume that ground input is able to resolve all failures. The plan is still pre-generated, but task failures can be handled without termination of execution, albeit in a costly manner. In the FE strategy, we allow flexible execution of our plans, which

provides another failure resolution mechanism. Flexible execution is less costly, but is able to handle only a fraction of possible failures, with all other failures handled by waiting for ground input. Finally, in the replan strategy, we allow for modification of the plan at execution time according to information discovered while running. This provides another failure resolution mechanism that we assume is more powerful the FE, but less powerful than ground input. In addition, replanning allows for the optimization of plans during execution time according to newly discovered utility. Replanning can therefore serve dual purposes: resolving task failures, and changing the plan to increase overall utility gain.

Given this context, we predict the overall utility achievement of a plan using an estimate of utility per unit cost u_{avg}. Then, assuming that tasks always succeed, our expected utility for a plan would be bu_{avg}. To factor in task failure, we assume that tasks fail with some probability $P(\text{fail})$, and we assume that task failures follow a Poisson distribution. The first planning/execution strategy that we analyze is the static strategy. Here, since the strategy terminates execution on failure, the system's expected utility achievement is based on how long it can be expected to execute. Then, the expected utility achievement of this strategy is given by:

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

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

In the ground strategy, we include a rudimentary error response of "going to ground" to seek manual intervention. To model this in our framework, we assume that such "wait for input" responses each incur a cost c_w, and always allow plan execution to continue. Then, if our plan has n_p tasks, the utility achievement of this "ground" strategy is:

$$U(S_g) = u_{avg}\,(b - P(\text{fail})n_p c_w) \tag{2}$$

In the FE strategy, we introduce flexible execution and assume that some subset of task failures can be resolved with this feature. We denote the probability of a task failing in this way as $P(\text{FE})$. Note that $P(\text{FE}) < P(\text{fail})$, since failures that are resolvable by flexible execution are a subset of all task failures in general. We assume that flexible execution has a negligible cost. Then, the utility achievement of plan execution using this strategy is:

$$U(S_f) = u_{avg}\,(b - (P(\text{fail}) - P(\text{FE}))n_p c_w) \tag{3}$$

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. We denote the probability that a given task fails in a way that can be resolved via replanning, but not flexible execution, as $P(\text{replan})$. Finally, we assume that all failure modes can be resolved via waiting for ground input. Then, if we denote the probability that a failure is resolvable only by ground input as $P(\text{wait})$:

$$P(\text{fail}) = P(\text{wait}) + P(\text{replan}) + P(\text{FE}) \tag{4}$$

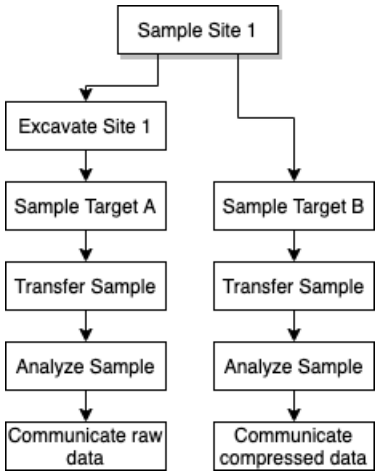

Figure 2: Two possible decompositions of a single parent "Sample Site 1". In the left decomposition, the lander excavates the site, samples target A, and communicates raw data. In the right decomposition, the lander skips excavation, samples site B, and communicates compressed data. Both achieve the same goal of sampling site 1.

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

$$\begin{aligned} U(S_r) =& du_d + u_{avg} \\ & (b - n_p(P(\text{wait})c_w - P(\text{replan})c_r)) \end{aligned} \tag{5}$$

Planning Approach

Problem Model

We model this problem using a hierarchical task network (HTN) to compile the domain-specific knowledge of the dependency structure into the task network. HTNs have been used successfully in industrial and other real-world applications to improve the tractability of planning problems in systems such as SHOP2 (Nau et al. 2003) and SHOP3 (Goldman and Kuter 2019). In an HTN, hierarchical tasks are decomposed to 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.

Decompositions provide a number of benefits to our planning approach, significantly reducing plan search space. In addition, we can treat all subtasks of a parent task as a singular block for planning purposes. The lander only 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, if for example, 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, which consists of post-landing initialization and other one-time initialization tasks. Second are sampling tasks. These consist of excavation, sample collection, transfer, analysis, and communication tasks. Excavation can take place at one of two excavation sites, and may be skipped if an excavation has previously occurred for the specified site. For collection tasks, the lander may choose between four collection targets: two for each excavation site. It may revisit a target that has already been sampled, still obtaining utility for a repeat sample. 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.

In our problem, 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.

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
            if edge.task not in new_plan and
              new_plan.cost + edge.cost below
              max_cost then
                add (new_plan, edge) to explore_q;
            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;
```

Planning Algorithm

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 branch and bound search on this graph and select the best plan explored. The algorithm consists of four phases: pre-processing, initialization, exploration, and plan selection.

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. 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 utility is the sum of each of its subtasks' utility. The same is true for energy cost. This step is performed once per domain model, offline. Preprocessing has exponential runtime in the worst case, and future work may require additional search in decomposing tasks as well as planning them.

Our search graph consists of nodes containing partial plans and their associated energy cost and utility. A node's cost is simply the sum of the costs of each task in the node's partial plan; the same goes for utility, though future work may take joint utility into account. In the initialization phase, the algorithm creates a single node containing an empty

plan, with utility and cost 0. Then, it iterates through all 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.

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.

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, 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. 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 a lower energy cost is preferred.

Empirical Evaluation

To test our model, we ran simulations of our planning and execution system on three variants of the Europa Lander domain described in Figure 1. The first is the base scenario. Here each task consumes an amount of energy that matches its a priori expectation in the task network, but may be noisy, 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.

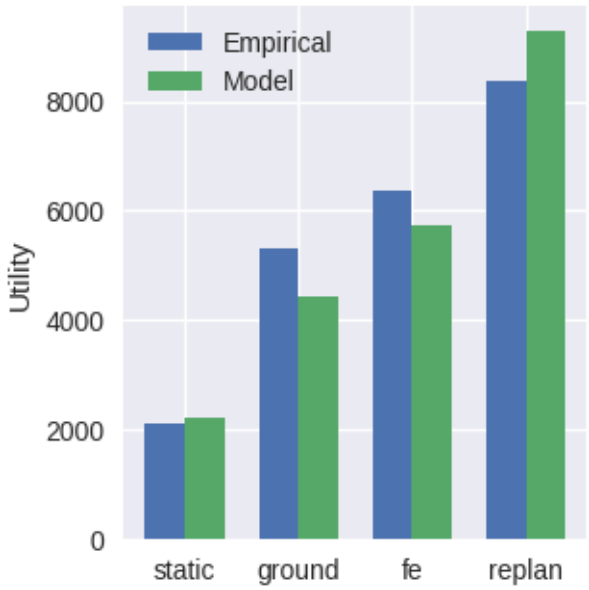

Figure 3: Average utility achieved in simulation of the base Europa Lander domain for 4 planning strategies, compared to theoretical model predictions.

For our model calculations, 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 0.1, and we assume flexible execution is able to handle 30% of such failures, while replanning is able to handle an additional 60% of failures. Thus, $P(\text{replan}) = 0.04$ and $P(\text{FE}) = .02$.

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. However, it 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 3, we compare the predictions of our model to the measured utility achievement of our system in simulation. We see that the four strategies follow the general contour of our model's predictions, but vary by some amount.

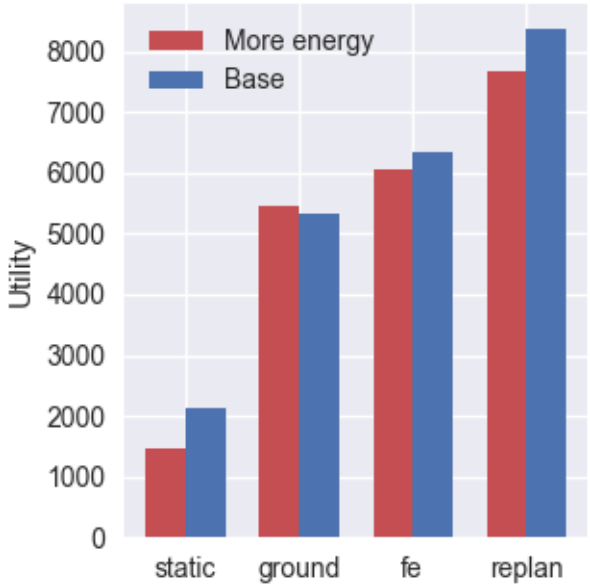

Figure 4: Average utility achieved in simulation of the Europa Lander domain where all tasks take 10% more energy than expected, compared to empirical results in the base domain.

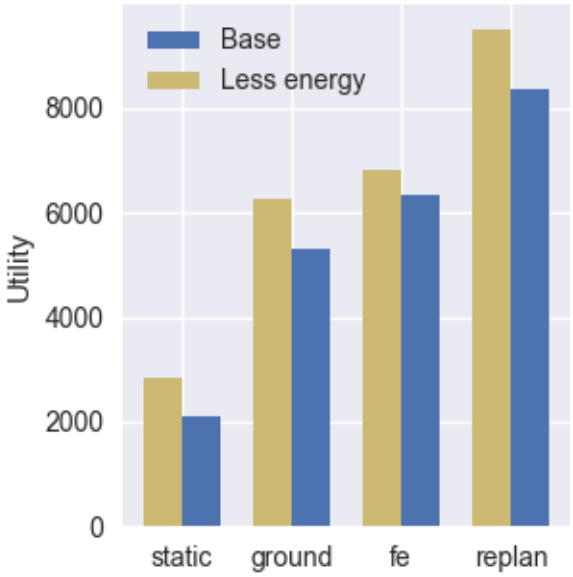

Figure 5: Average utility achieved in simulation of the Europa Lander domain where all tasks take 10% less energy than expected, compared to empirical results in the base domain.

While our predictive model generally matches our empirical measurements, it is limited in some aspects. 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 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 addition, in the Europa Lander domain, sites only need to be excavated a single time, and 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 tends to halt while repeatedly sampling from an existing site, the model underestimates utility. This behavior is prominently seen in the ground and FE strategies in Figure 3. 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.

For the replan strategy in particular, 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 and 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 gain of 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 4. 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 appears generally 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 5). 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 ex-

tra 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 replan strategy is more often able to process a sample.

Related Work

Decision-theoretic planning is an effective approach to planning under uncertainty, particularly in robotic domains, as it provides a formal model for reasoning about problems in which actions have stochastic outcomes or the agent has incomplete information about its environment (Iocchi et al. 2016; Saisubramanian, Zilberstein, and Shenoy 2017; Zilberstein et al. 2002). The primary objective of decision-theoretic planning is to produce plans or policies that define the potential trajectories of actions that the agent may take which maximizes its expected utility, rather than maximizing or guaranteeing goal-reachability (Boutilier, Dean, and Hanks 1999). A standard approach in decision-theoretic planning for modeling domains is to use a Markov decision process (MDP) (Bellman 1957) when the agent knows the full evaluation of every state at each timestep, or a partially observable Markov decision process (POMDP) (Spaan 2012) where this holds only for a subset of the variables that define the statespace.

However, several issues in spacecraft or rover operations complicate the use of said decision making models. First, these models traditionally do not support durative or concurrent actions, but rather assume that all actions are instantaneous and fully sequential in nature. Second, although there have been a number of approaches over the years aimed at improving the scalability of these approaches (Guestrin et al. 2003; Wray, Witwicki, and Zilberstein 2017; Yoon, Fern, and Givan 2007), most algorithms that solve MDPs produce policies that account for all contingencies and provide actions for all states in the domain. This is generally impractical or impossible in spacecraft and rover operations where computational power is (often severely) limited, and more so in our problem where the battery is non-rechargeable and the domain model is expected to be modified repeatedly throughout the agent's operation.

Onboard planning and execution are of great interest to the space domain. Flexible execution of tasks is a central focus of execution engines like PLEXIL (Verma et al. 2005) and TRACE (de la Croix and Lim 2020). The Earth Observing One (EO-1) spacecraft (Chien et al. 2005), which flew for over 12 years from 2004-2017, was designed specifically to react to dynamic scientific events. Planning was performed by the CASPER planning software (Chien et al. 2000), which took on the order of 10s of minutes to replan but did not produce temporally flexible plans. To address this, the onboard executive (SCL) was able to flexibly interpret the *execution* of a plan to handle minor execution runtime variations. The flight and ground planners (Chien et al. 2010) both used a domain specific search algorithm that enforced a strict priority model over observations for a limited model of utility. Recently, the Intelligent Payload Experiment (IPEX) also successfully used the CASPER planning software to achieve its mission objective, further validating the efficacy of using onboard replanning to handle dynamic events and observations during operation even when the plans are not temporally flexible (Chien et al. 2017).

The M2020 Perseverance rover also plans to fly an onboard planner (Rabideau and Benowitz 2017) to reduce lost productivity from following fixed time conservative plans (Gaines et al. 2016). Like the planning approach we propose in this paper, the M2020 planning architecture also relies on rescheduling and flexible execution (Chi et al. 2018), ground-based compilation (Chi et al. 2019), heuristics (Chi, Chien, and Agrawal 2020), and very limited handling of planning contingencies (Agrawal et al. 2019). However, it uses a non-backtracking planner, which cannot take advantage of plan optimization or utility discovery. Our work also takes a different focus, primarily examining the effects of task failure and considering integrated planning in the context of failure resolution. Finally, many characteristics of the M2020 mission are fundamentally different from the mission concept we consider here, such as the lack of reliable a priori model parameters, the inability to recharge the battery, and the long communications blackout time windows incentivizing greater mission autonomy.

Future Work

Our work focuses primarily on our planning system's response to adverse events such as task failure. Our examination of positive exogenous events is limited to analysis of utility discovery. However, in space exploration domains, due to conservative parameter assignments, we often find that tasks finish early or use fewer resources than the margin allocated to them. Reasoning about these events may provide a model that more accurately represents the reality of the Europa Lander domain.

In addition, in this work we focus primarily on energy as a resource. However, a number of other resources exist, and the consumption of any of these may be noisy or biased, affecting plan execution. In particular, task execution time has wide-ranging effects on both task energy use and plan execution as a whole, especially when deadlines come into play. These deadlines are especially present in the Europa Lander domain in the form of ground communication windows. Task execution time and other variables therefore represent a significant unexplored area of work in this domain.

While we react to uncertainty at execution time, we do not take this into account when planning. This is apparent in our system's behavior in the scenario where tasks take more energy than expected. A more sophisticated planner would explicitly integrate probability of such adverse events, maximizing expected utility. For example, excavation tasks involve risk; task failure could result in significant energy loss or damage to the lander. Reasoning about exogenous events such as these would improve utility achievement by potentially avoiding such risks, or even seeking them out later in the mission when failure is less impactful.

Acknowledgments

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Agrawal, J.; Chi, W.; Chien, S.; Rabideau, G.; Kuhn, S.; and Gaines, D. 2019. Enabling Limited Resource-Bounded Disjunction in Scheduling. In *11th International Workshop on Planning and Scheduling for Space (IWPSS 2019)*, 7–15.

Bellman, R. 1957. A Markovian decision process. *Journal of Mathematics and Mechanics* 679–684.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)* 11:1–94.

Chi, W.; Chien, S.; Agrawal, J.; Rabideau, G.; Benowitz, E.; Gaines, D.; Fosse, E.; Kuhn, S.; and Biehl, J. 2018. Embedding a Scheduler in Execution for a Planetary Rover. In *International Conference on Automated Planning and Scheduling (ICAPS 2018)*.

Chi, W.; Agrawal, J.; Chien, S.; Fosse, E.; and Guduri, U. 2019. Optimizing Parameters for Uncertain Execution and Rescheduling Robustness. In *International Conference on Automated Planning and Scheduling (ICAPS 2019)*.

Chi, W.; Chien, S.; and Agrawal, J. 2020. Scheduling with Complex Consumptive Resources for a Planetary Rover. In *International Conference on Automated Planning and Scheduling (ICAPS 2020)*.

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

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davis, A.; Mandl, D.; Frye, S.; Trout, B.; et al. 2005. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2(4):196–216.

Chien, S.; Tran, D.; Rabideau, G.; Schaffer, S.; Mandl, D.; and Frye, S. 2010. Timeline-based space operations scheduling with external constraints. In *Twentieth International Conference on Automated Planning and Scheduling*.

Chien, S.; Doubleday, J.; Thompson, D. R.; Wagstaff, K. L.; Bellardo, J.; Francis, C.; Baumgarten, E.; Williams, A.; Yee, E.; Stanton, E.; et al. 2017. Onboard autonomy on the intelligent payload experiment cubesat mission. *Journal of Aerospace Information Systems* 14(6):307–315.

de la Croix, J.-P., and Lim, G. 2020. Event-driven modeling and execution of robotic activities and contingencies in the europa lander mission concept using bpmn. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*.

Gaines, D.; Anderson, R.; Doran, G.; Huffman, W.; Justice, H.; Mackey, R.; Rabideau, G.; Vasavada, A.; Verma, V.; Estlin, T.; et al. 2016. Productivity challenges for Mars rover operations. In *Proceedings of 4th Workshop on Planning and Robotics (PlanRob)*, 115–125. London, UK.

Goldman, R. P., and Kuter, U. 2019. Hierarchical Task Network Planning in Common Lisp: the case of SHOP3. In *Proceedings of the 12th European Lisp Symposium*, 73– 80. Zenodo.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)* 19:399–468.

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

Iocchi, L.; Jeanpierre, L.; Lazaro, M. T.; and Mouaddib, A.-I. 2016. A practical framework for robust decision-theoretic planning and execution for service robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Nau, D. S.; Au, T. C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20:379–404.

Rabideau, G., and Benowitz, E. 2017. Prototyping an Onboard Scheduler for the Mars 2020 Rover. In *International Workshop on Planning and Scheduling for Space (IWPSS 2017)*.

Saisubramanian, S.; Zilberstein, S.; and Shenoy, P. 2017. Optimizing electric vehicle charging through determinization. In *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Scheduling and Planning Approaches*.

Spaan, M. T. 2012. Partially observable Markov decision processes. In *Reinforcement Learning*. Springer. 387–414.

Verma, V.; Estlin, T.; Jónsson, A.; Pasareanu, C.; Simmons, R.; and Tso, K. 2005. Plan execution interchange language (plexil) for executable plans and command sequences. In *International symposium on artificial intelligence, robotics and automation in space (iSAIRAS)*.

Verma, V.; Gaines, D.; Rabideau, G.; Schaffer, S.; and Joshi, R. 2017. Autonomous Science Restart for the Planned Europa Mission with Lightweight Planning and Execution. In *International Workshop on Planning and Scheduling for Space (IWPSS 2017)*.

Wray, K. H.; Witwicki, S. J.; and Zilberstein, S. 2017. Online decision-making for scalable autonomous systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 7, 352–359.

Zilberstein, S.; Washington, R.; Bernstein, D. S.; and Mouaddib, A.-I. 2002. Decision-theoretic control of planetary rovers. In *Advances in Plan-Based Control of Robotic Agents*. Springer. 270–289.