A Sampling-Based Optimization Approach to Handling Environmental Uncertainty for a Planetary Lander

Connor Basich,[1,2] **Daniel Wang,**[1] **Joseph A. Russino,**[1] **Steve Chien,**[1] **and Shlomo Zilberstein**[2]

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, {firstname.lastname}@jpl.nasa.gov
[2]University of Massachusetts Amherst, Amherst, Massachusetts, {cbasich, shlomo}@cs.umass.edu

Abstract

Planning for autonomous operation in unknown environments poses a number of technical challenges. The agent must ensure robustness to unknown phenomena, unpredictable variation in execution, and uncertain resources, all while maximizing its objective. These challenges are exacerbated in the context of space missions where uncertainty is often higher, long communication delays necessitate robust autonomous execution, and severely constrained computational resources limit the scope of planning techniques that can be used. We examine this problem in the context of a Europa Lander concept mission where an autonomous lander must collect valuable data and communicate that data back to Earth. We model the problem as a hierarchical task network, framing it as a utility maximization problem constrained by a monotonically decreasing energy resource. We propose a novel deterministic planning framework that uses periodic replanning and sampling-based optimization to better handle model uncertainty and execution variation, while remaining computationally tractable. We demonstrate the efficacy of our framework through simulations of a Europa Lander concept mission in which our approach outperforms several baselines in utility maximization and robustness.

Introduction

Planning in domains with large uncertainty and very low margins for error has long been a challenge in the AI planning community. While numerous techniques have been developed over the years, many are rendered impractical or infeasible by the technical constraints that many physical robotic systems face. In space-based scenarios, the uncertainty is often higher, the margins for error lower, and the constraints more severe. Traditional approaches to planning in space-based domains have consequently utilized deterministic or sampling-based planning methods (Chi et al. 2019; Chi, Chien, and Agrawal 2020) which are fast and computationally inexpensive, and can be very effective when paired with a priori expert domain knowledge. Recent work has investigated how periodic replanning, flexible execution, and online model updates can be used in conjunction with search-based deterministic planning to improve the efficacy and robustness of the overall plans generated and executed by a space-based robotic system (Wang et al. 2020).

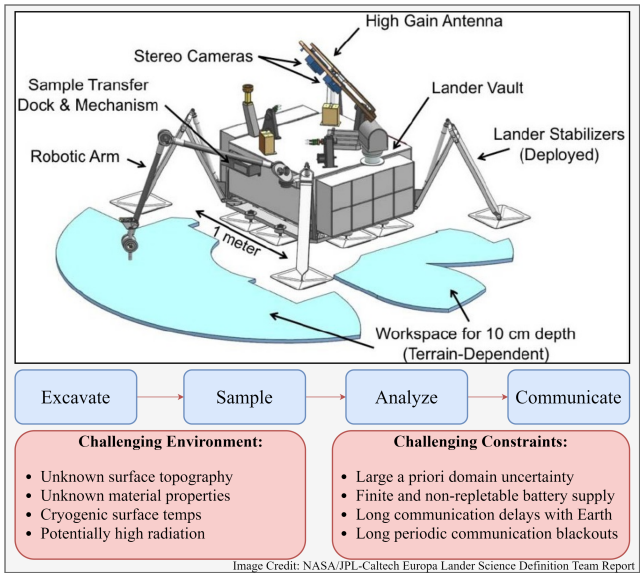

Figure 1: An illustration of the Europa Lander concept mission in which an autonomous lander is tasked with performing *in-situ* analysis of sampled surface material and communicating the collected data to Earth.

While often effective in practice, these approaches do not actively consider model uncertainty and off-nominal behavior during plan generation, and are hence *reactive* in how they handle the uncertainty faced by the system over its environment. In this work, we propose a planning framework that extends the algorithm from (Wang et al. 2020) by *proactively* anticipating deviations from nominal execution by incorporating domain uncertainty into the plan generation, creating plans that are more robust to these deviations without sacrificing solution quality in the nominal case. We examine the efficacy of our approach in the context of a proposed concept mission in which a lander is tasked with analyzing surface material to acquire valuable scientific data by performing *in-situ* analysis of samples excavated from the surface of the Jovian moon Europa, and communicating that data back to Earth (Hand 2017).

This concept mission entails several challenges that differentiate it from prior missions. First, a priori knowledge of the environment is severely limited and uncertain. Second,

the system's battery supply is finite and non-repletable (i.e. there is no possible power generation). Third, communication with Earth is constrained by two factors: (1) due to the large distance to Earth, there are long communication delays when communicating with Earth. As a result, ground-in-the-loop operations cannot be relied on as the system will be losing battery while waiting for communications and hence the lander must be capable of operating fully autonomously. And (2), due to Jovian occlusion, the lander will be faced with long periodic communication blackouts (roughly 42 out of every 84 hours) which constrain when the lander is capable of downlinking the data it has collected to Earth.

As utility is only assigned to data that is acquired and *successfully downlinked to Earth*, and none for data collected but not downlinked, in order for the Lander to be successful it needs to carefully manage the trade-off between data acquisition and communication. Furthermore, it must do this while constrained by a finite and monotonically decreasing battery supply, limited knowledge of its environment, and limited communication with Earth. As a result, for the mission to be successful, the system requires an autonomous planning and execution framework that is (1) computationally efficient; (2) robust to unprecedented levels of uncertainty; but still (3) capable of maximizing overall utility.

We model the problem as a *hierarchical task network* (HTN) (Nau et al. 2003) due to the structured nature of the tasks that the lander can perform, and consequently use an anytime heuristic-search algorithm designed for solving HTNs (Wang et al. 2020) as the primary subroutine of the proposed approach. Our approach is based on principles from Hindsight Optimization (HOP) (Chong, Givan, and Chang 2000) and works by hypothesizing a set of sampled scenarios that the system may face, planning for each scenario, and evaluating each of the sampled plans' performances across all scenarios. The plan with the highest weighted value is selected (this can be viewed as an approximation to maximizing expected utility). As the planner is deterministic, we also perform periodic replanning to ensure that the system's performance does not deviate too far from expectation. This approach has similarities with determinization-based methods that have been highly successful in solving large Markov decision processes (MDPs) (Yoon, Fern, and Givan 2007; Yoon et al. 2008; Pineda and Zilberstein 2014). While MDPs and other stochastic sequential decision making models have been had success in many settings, they are computational expensive and ill-suited for domains with concurrent actions and continuous states such as that which is considered in this paper.

We empirically validate our approach in a simulated Europa-like concept mission. The execution system we use in our simulations is MEXEC, an integrated planner and executive first built for NASA's Europa Clipper mission (Verma et al. 2017), to better react to variations in both environment and execution. Finally, to compensate for both uncertain model priors and the deterministic nature of the planner, the framework replans on a periodic basis. We present empirical results against two base-line approaches similar to those used in prior missions: a greedy planner with replanning and an anytime heuristic-search based planner with replanning. We show that the proposed planning framework, **HTNSearch-PHRA**, is more effective on average and more robust to uncertainty across five different mission scenarios. In addition, we analyze the effect of increasing the size of the hypothesized scenario set by comparing the performance of the algorithm with four different sets of hypothesized scenarios.

Problem Description

Domain Overview

The primary goal of the Europa Lander concept mission is to excavate and sample the moon's surface, analyze the sampled material for signs of biosignatures, and communicate that 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 therefore limited to primary objective tasks, secondary objective tasks, and data communication. This provides significant structure to the problem, since the concept mission clearly defines the sequence of actions required to achieve these goals. Figure 2 displays an example of a potential execution trace of tasks that satisfies the minimum requirements of the mission, and illustrates the inherent structure of the concept mission amongst the possible tasks.

As a minimum requirement, the lander should excavate a trench in the Europan surface, collect three samples from that site, analyze those samples, and communicate 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. In addition, the lander may choose to resample the same location to, for example, verify the discovery of a biosignature at that location. In the baseline concept mission, 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. Because communication is difficult and energy intensive, the lander may choose to compress data lossily which reduces both the energy required to communicate the data and its utility, if the expected utility of this action is higher.

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.

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

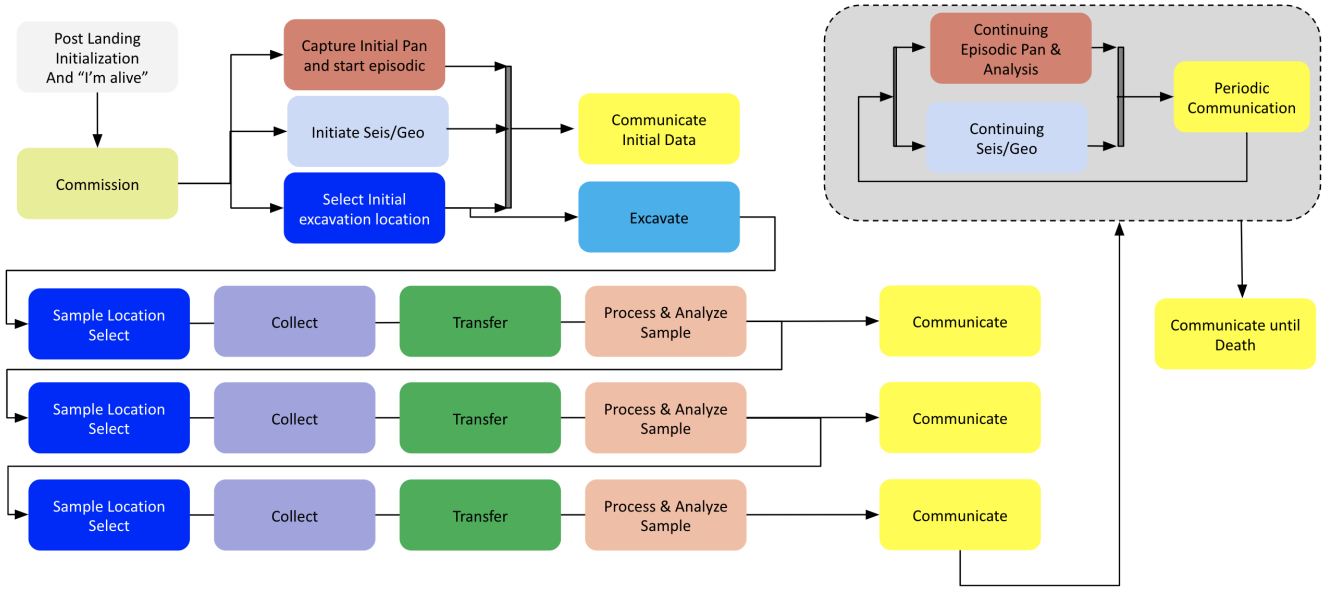

Figure 2: An illustration of a potential execution trace of an example task network for the Europa Lander concept mission that satisfies the minimum requirements of the mission.

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 concept mission 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 – including sleeping – consumes energy from the battery, and the algorithm must plan accordingly to maximize utility in the 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 are assumed to be inaccurate. 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.

Problem Formulation

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 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 partially ordered 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 enable us to significantly reduce the planning search space as we can treat all subtasks of a parent task as a singular block during the planning process; for example, the model treats "excavate, sample, transfer, analyze" as a single unit and schedules the subtasks accordingly.

Formally, an input to the problem is a set $\mathcal{T} = \{T_1, ..., T_n\}$ and a system state S. Each T_i is a task that is represented by the tuple $\langle p_i, d_i, e_i, u_i, s_i, C_i, W_i, D_i \rangle$ where p_i is the priority of the task, d_i is the expected duration of the task, e_i is the expected rate of energy usage by the task, u_i is the expected utility of the task, s_i is the preferred start time of the task, C_i is the set of constraints that must be satisfied for the task to be scheduled, $W_i = \{[t_{i1}, t_{i2}], ..., [t_{in-1}, t_{in}]\}$ is the set of time windows that the task can be scheduled in, and $D_i = \{T_{i1}, ..., T_{im}\}$ is the partially ordered set of decompositions of the task, which can be empty if T_i is a primitive task. The state S is represented by a collection of continuous and discrete features including the remaining battery supply, the current data load, the current time, and all features required to model task constraints.

There are four main parent task types in the mission model. The first is a `Preamble` which consists of post-landing initialization and other one-time initialization tasks, and must be executed immediately upon landing. Second are data acquisition tasks which consist of `excavation`, `sample collection`, `sample transfer`, and `sample analysis` tasks. Excavation can take place in one of three excavation sites, and may be skipped if a site has previously been excavated. For collection tasks, the lander may choose between one of sev-

eral collection targets at any given excavation site (repeated sampling of the same target is allowed with no penalty). The analysis task returns the dataload acquired from a given sample upon completion; dataload represents the maximum potential utility that the acquired data provides upon successful downlink to Earth. Third, there are Seismograph/Panorama tasks which consist of `seismographic data collection` and `panoramic image data collection`; these tasks provide less data but are more reliable in their execution. Fourth is the `communication` task which decomposes into either a single, or a sequence of two, communication(s), either of which can be of raw data or compressed data. In this problem, we assign utility solely to the successful completion of a communication task, where communicating raw data provides greater utility but consumes more energy than communicating compressed data; note that both communication tasks consume the same amount of dataload. Utility is assigned to tasks *a priori* but we note that in practice may likely be updated online as new information is obtained by the system.

Approach

The underlying planning method employed in this approach relies on heuristic search. Search-based planning algorithms have been popular for a number of years as they (1) do not require that the full state space is evaluated to produce a solution (Hansen and Zilberstein 2001), (2) are often anytime algorithms that can return a solution at any point during runtime (Zilberstein 1996), and (3) can easily leverage heuristics to reduce the computational burden while still achieving high performance (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Korf 1990). In this case, all three of these properties are highly desirable, and influenced our decision to utilize a heuristic search-based planning algorithm.

Heuristic HTN Search

The main planning algorithm, Algorithm 1, relies on the heuristic search approach for solving HTNs described in (Wang et al. 2020) – which we henceforth refer to as `HTNSearch` – as its primary subroutine. We offer a brief discussion of their algorithm.

`HTNSearch` first performs a pre-processing step in which all task decompositions are flattened into a single layer so that parent tasks are simply linear chains of primitive tasks. By flattening the decompositions, we can assign specific utility and energy values to each parent task as there is no ambiguity over which decomposition it is represented by. Note that this step can be performed offline and only needs to be performed once.

Next, `HTNSearch` initializes the search graph on the newly flattened task network, where nodes hold partial plans and edges hold (flattened) task decompositions or primitive tasks. As the algorithm is deterministic, both the cost and utility associated with any node is the sum over the tasks in the partial plan for the respective value. Finally, the algorithm performs a heuristic branch and bound search procedure over the search graph where the search is bounded by both the feasibility of partial plans (their total energy cost cannot exceed the current battery supply, and the plan adheres to inter-task constraints), and optional computational constraints on the number of explorable nodes. For any plan and decomposition pair, (P, d), a density based heuristic value of $utility(P) + \frac{utility(d)}{cost(d)}$ is used and ties are broken in favor of lower cost.

The main limitation of this approach, which drives the motivation for Algorithm 1, is that it is a deterministic algorithm for a non-deterministic domain. In other words, the plan that is produced assumes that the future will operate exactly according to expectation. (Wang et al. 2020) address this issue primarily through the use of online model updates and frequent replanning to 'course-correct' the system online. This idea is similar to that of determinization-based approaches for solving very large Markov decision processes, such as FF-Replan (Yoon, Fern, and Givan 2007), which have been shown to perform well in the MDP setting, particularly in the infinite horizon case, but are not robust to dead-ends. Hence, we propose a planning algorithm that *proactively* considers off-nominal behavior and execution during plan time, rather than just *reactively* responding to off-nominal behavior and execution. We demonstrate through empirical evaluations that our approach is indeed more robust to off-nominal scenarios than the standard heuristic search, performing as well or better in both positive and negative scenarios.

HTNSearch with Post Hoc Robustness Analysis

The proposed planning algorithm, the pseudocode for which can been seen in Algorithm 1, is intuitively straightforward. The algorithm takes in as input an instance of the Europa Lander problem modeled as a hierarchical task network, $\mathcal{T}$, and begins by producing a set of hypothesized scenarios via the subroutine *hypothesizeScenarios* (line 3). The function *hypothesizeScenarios* takes in the current task network that is being planned on and the system's current state, and returns a set of hypothesized scenarios $\mathcal{H}$. A scenario, $h \in \mathcal{H}$, is comprised of an initial state and an instantiation of the parameterized domain. This function is left general as its implementation will be both domain and purpose specific. For example, in our simulations we hypothesize multiple scenarios where the current battery life of the lander is varied up or down to represent the uncertainty over the "true" battery life of the lander. However, other parameters may be varied such as the time or energy to perform various tasks, the likelihood of positive data from different samples, or the possibility of excavated sites collapsing. These scenarios may be developed using expert knowledge a priori, or may be generated online by drawing from distributions that parameterize the domain model.

For each hypothesized scenario $h \in \mathcal{H}$, the algorithm first instantiates h by updating the relevant parameters of $\mathcal{T}$, and calls `HTNSearch` on the newly instantiated task network to produce the best-found plan $\mathcal{P}$ (lines 5-7) within the computational constraint. The plan $\mathcal{P}$ is evaluated on each scenario $h' \in \mathcal{H}$ by computing the expected utility following $\mathcal{P}$, $V^{\mathcal{P}}(\mathcal{T}, h')$ (line 10) using a stochastic execution graph subroutine. The observed expected utility is weighted by a

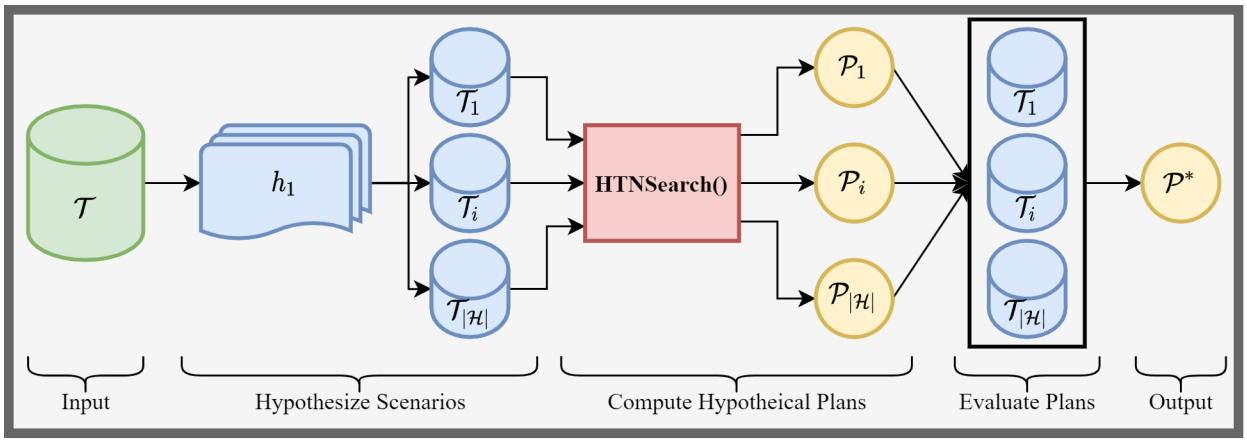

Figure 3: An illustration of the HTNSearch-PHRA algorithm. A tasknet, $\mathcal{T}$, and a set of hypothesized scenarios, $h_1, ..., h_{|\mathcal{H}|}$, produce set of instantiated task networks, $\mathcal{T}_1, ..., \mathcal{T}_{|\mathcal{H}|}$. Each $\mathcal{T}_i$ is comprised of the input task network with certain parameters modified by the scenario; for instance the current state of the system or the cost and utility of various tasks. The planning algorithm, HTNSearch, is ran on each $\mathcal{T}_i$ to produce a plan, $\mathcal{P}_i$ that is the best plan found for that instantiated task network. Each plan, $\mathcal{P}_i$ is evaluated on all $\mathcal{T}_i$ to produce a cumulative score that is comprised of a weighted sum of expected utilities, where the weights are determined by the likelihood of the hypothesized scenario. The plan that has the highest score, $\mathcal{P}^*$, is returned.

function *getScenarioWeight* and the weighted value is added to the total score of the plan, $\mu^{\mathcal{P}}$ (9-10). Finally, the plan that had the highest total score is returned.

The function *getScenarioWeight* returns a real valued number in $[0, 1]$, given an HTN and a scenario. In our case, we compute the Gauss-Legendre quadrature weights assuming that the parameters relevant to the hypothesized scenarios are normally distributed. In general we believe that any relative likelihood-based weighting scheme will work, however we note that offline optimization of these weights may be worthwhile particularly when the scenarios considered are over multiple different task network parameters.

We compute the expected utility of the plan $\mathcal{P}$ using a stochastic subroutine that builds the non-deterministic execution graph of $\mathcal{P}$ given the distributions which parameterize the task network (energy cost, duration, data, and utility). Computing the expected utility of a deterministic plan in a stochastic domain is significantly cheaper than computing a fully stochastic policy in the first place, and still allows us to observe a more accurate evaluation of each plan.

Finally, as the HTN search algorithm dominates the other subroutines in terms of runtime complexity, the runtime of Algorithm 1 is $\sim |\mathcal{H}|$ times the runtime of HTNSearch when $|\mathcal{H}|$ is small. However, as $|\mathcal{H}|$ grows, the computation spent evaluating plans grows at a rate of $O(|\mathcal{H}|^2)$. Furthermore, there are diminishing returns to increasing the number of hypothesized scenarios considered, particularly when adding very low likelihood scenarios to $\mathcal{H}$. An analysis of this can be found further in the paper. Ultimately, the problem of determining which, and how many, scenarios to include in $\mathcal{H}$ is an important element in balancing the trade off between efficiency and effectiveness.

Algorithm 1: HTNSearch-PHRA

Input: A hierarchical task network $\mathcal{T}$
Result: A plan $\mathcal{P}^*$

1 $\mathcal{P}^* \leftarrow$ None
2 $\mu^* \leftarrow -\infty$
3 $\mathcal{H} \leftarrow hypothesizeScenarios(\mathcal{T}, s)$
4 **for** $h \in \mathcal{H}$ **do**
5 $\quad \hat{\mathcal{T}} \leftarrow instantiateScenario(\mathcal{T}, h)$
6 $\quad \mathcal{P} \leftarrow HTNSearch(\hat{\mathcal{T}})$
7 $\quad \mu^{\mathcal{P}} \leftarrow 0$
8 $\quad$ **for** $h' \in \mathcal{H}$ **do**
9 $\quad\quad \gamma \leftarrow getScenarioWeight(\mathcal{T}, h')$
10 $\quad\quad \mu^{\mathcal{P}} \leftarrow \mu^{\mathcal{P}} + \gamma V^{\mathcal{P}}(\mathcal{T}, h')$
11 $\quad$ **end**
12 $\quad$ **if** $\mu^{\mathcal{P}} > \mu^*$ **then**
13 $\quad\quad \mathcal{P}^* \leftarrow \mathcal{P}$
14 $\quad\quad \mu^* \leftarrow \mu^{\mathcal{P}}$;
15 $\quad$ **end**
16 **end**
17 **return** $\mathcal{P}^*$

Experiments

Experimental Setting

To evaluate the performance of Algorithm 1, we compared against two baselines: HTNSearch and GREEDY. In GREEDY, priorities were assigned a priori to each task decomposition, and the planner greedily attempts to schedule tasks in order of priority at each planning cycle, skipping over tasks if they cannot be scheduled due to conflicts or violated constraints. Priorities are assigned offline using a com-

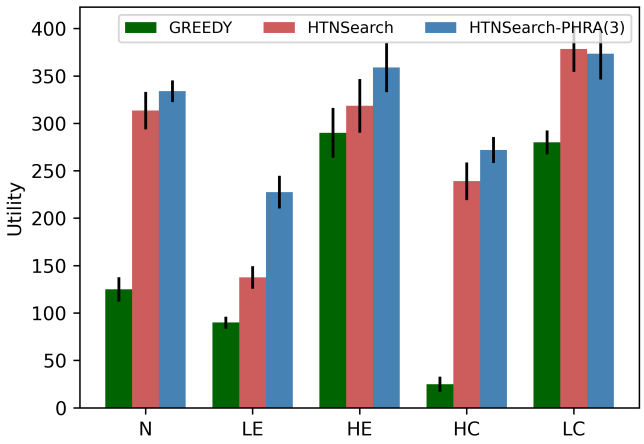

Figure 4: Utility achieved by each planning algorithm on all five scenarios. Values shown are mean and standard error over 10 trials.

bination of hard-coded domain knowledge (e.g. the Preamble must have the highest priority) and Monte carlo trials on sampled priority orderings across the input task.

The domain used for our simulations is described in Section 2. We recall a few key points here. First, the system is tasked with excavating the surface of the moon, collecting samples to analyze which produces data, and then communicating that data back to Earth. Only data that has been successfully communicated provides utility. Second, the agent's battery supply is consumptive and *non-rechargeable*, and the system maintains a base Hotel load – a constant energy usage – even when sleeping. Third, communication with earth is only possible in cycles due to Jovian occlusion (every other 250 time units in our simulations). Consequently, the system must be able to effectively manage a monotonically decreasing battery supply and fixed time windows for communication, while still performing tasks that produce data, to actually receive any utility. Furthermore, it must execute these tasks in a domain with large a priori uncertainty.

We therefore considered 5 different stochastic scenarios: (1) Nominal, (2) Low Energy, (3) High Energy, (4) High Consumption, (5) Low Consumption. In scenario (1) there are no off-nominal, or unexpected, events or behaviors that occur during simulation. In scenarios (2) and (3) there is a sudden change ($\pm 20\%$) in remaining battery life that occurs 500 time units into the simulation as the battery recalibrates. In scenarios (4) and (5) energy is stochastically drained or added to the observed remaining battery supply at every state update. In all scenarios, task parameters such as energy rates, duration, and data load are all drawn from low variance Gaussians centered around pre-determined means at runtime. Each scenario was simulated 10 times for each planning algorithm to account for this execution variation.

In these experiments, we specifically focused on off-nominal variations on remaining battery for three reasons. The first is that, historically, battery measurements have come with large uncertainty. The second is that battery is the most valuable resource in this domain, as time only matters in that there is a constant minimum Hotel load, and zero remaining battery supply is a terminal absorbing state. The third reason is that deviations in battery supply or energy consumption act as direct proxies for most other off-nominal behavior (extra time to complete a task, getting stuck, failing to perform a task requiring it to be repeated, etc.). However, the algorithm presented is not relegated to such a constraint, and in general can capture arbitrary scenarios.

Experimental Results

The main results of our experiments can be seen in Figures 4 and 5. The first experiment compares the performance of three algorithms: `GREEDY`, `HTNSearch`, and `HTNSearch-PHRA(3)`. The second experiment compares the performance of four variations of `HTNSearch-PHRA`, where the size of the hypothesized scenario set is increased: `HTNSearch-PHRA(1)`, `HTNSearch-PHRA(3)`, `HTNSearch-PHRA(5)`, and `HTNSearch-PHRA(7)`.

Cross-Method Comparison Figure 4 shows the mean and standard error of the utility achieved by each planning algorithm across the five planning scenarios. `GREEDY` performs well when the scenario is an optimistic scenario as the system ends up with enough battery to perform all of the high priority tasks without issue; in particular this means sampling more than the other approaches as `GREEDY` does not perform the more conservative actions such as Seis/Pan which produce less data but are more stable tasks. However, in the pessimistic scenarios, where the battery supply ends up being less than expected, `GREEDY` performs extremely poorly, using up too much energy early on in the mission sampling new targets to collect data, leaving insufficient energy to communicate all of the data back. This demonstrates the brittleness of a greedy approach in the context of a domain with large uncertainty.

`HTNSearch` performs well overall, and notably better than `Greedy`, as expected based on the results from Wang et al. (Wang et al. 2020). However, because its plans are always conditioned on nominal behavior and a lack of unexpected events, it is always reacting to off-nominal deviations, leading to poorer results in both the Low Energy (LE) scenario and the High Consumption (HC) scenario. Notably, the performance is better in the High Consumption scenario, where the negative impact on on the battery supply is constant but small, than in the Low Energy scenario where the impact is large, sudden, and unexpected. The former scenario allows for constant reactive replanning to work as a viable strategy for managing the off-nominal performance, but if too much energy has been spent prior to encountering the sudden drop in battery supply in the latter scenario, there is no recourse for the system. The performance in the optimistic scenarios, High Energy and Low Consumption, indicate that the system is able to communicate an extra dataload on average in the low consumption case. Similar to above, we believe that the reason for this is that the system can constantly react to the small increases in battery supply in the Low Consumption scenario, but cannot plan to take advantage of the "excess" energy it encounters in the High Energy case.

On the other hand, `HTNSearch-PHRA`, which proac-

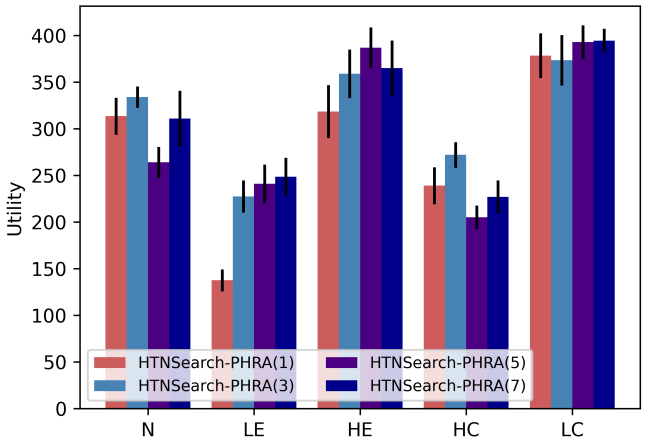

Figure 5: Utility achieved by Algorithm 1 with $|\mathcal{H}| = 3, 5,$ and 7 respectively, on all five scenarios. Values shown are mean and standard error over 10 trials.

tively selects plans that are robust to low-energy hypothesized scenarios, is more robust to the negative conditions, producing more utility on average in both the Low Energy and High Consumption scenarios. The difference here is more significant in the Low Energy scenario as `HTNSearch-PHRA` can proactively plan for having less energy, and does not just react to the latest observations and state updates. However, the performance is still greater in the High Consumption scenario because it can constantly respond to the small off-nominal deviations in behavior, and though it selects plans that are robust to high energy scenarios, it does not know that such a scenario will occur until it has. In the optimistic scenarios where energy is more abundant than expected, both `HTNSearch` and `HTNSearch-PHRA` perform comparably as they are both able to make use of the excess battery supply, having achieved higher utility than in the Nominal scenario. As above, however, our algorithm performs better in the High Energy case as it proactively creates plans robust to similar situations. Finally, it is worth emphasizing that `HTNSearch-PHRA` also performed comparably – up to noise – to `HTNSearch` in the Nominal scenario where the base task network that is used by `HTNSearch` is correct (up to stochasticity). This means that our approach, although sensitive to off-nominal scenarios, does not sacrifice performance quality in the nominal case as well.

Overall, these results demonstrate that the proposed algorithm performs comparably or better to each baseline approach in all scenarios tested, but the benefits are most notable in scenarios where the deviations are large and sudden, rather that small and frequent, as both search-based algorithms have the ability to respond to the latter events via replanning in order to "course correct", and in pessimistic scenarios where off-nominal behavior hurts performance rather than aids. This is because the *reactive* approach can always benefit from extra battery supply as the excess is observed, but can not always bounce back from energy deficits. However, proactively producing plans that are sensitive to both these scenarios ensures that the system follows a plan that never performs too poorly in any hypothesized scenario, while also retaining the benefits of reactively course-correcting.

Analysis of $\mathcal{H}$ on the Quality of `HTNSearch-PHRA` If we consider the results from Figure 5, we observe that increasing the size of the hypothesized scenario set can lead to improved performance, but not always and not to a large extent (up to noise). We suggest that the reason for this is that hypothetical scenarios with low likelihood – particularly in this case where none of the scenarios consider critical failures – impact the score of each generated plan to a sufficiently low degree that plans generated for low likelihood scenarios are never actually selected to be scheduled. This issue may be compounded by the fact that each hypothesized scenario alters the same parameter, namely battery supply.

The reason that `HTNSearch-PHRA(5)` and `HTNSearch-PHRA(7)` perform better than `HTNSearch-PHRA(3)` in the energy based scenarios compared to the consumption based scenarios is likely that the ability to constantly course correct in response to small observed deviations dominates the effect of considering low-likelihood scenarios during the planning phase. However, if we consider scenarios (N) and (HC), `HTNSearch-PHRA(5)` and `HTNSearch-PHRA(7)` actually perform worse than the other two. The reason for this is that the plans selected are too conservative – on account of being scored on low likelihood negative outcomes – and end up costing utility over the course of the full problem horizon.

In the future, we plan to analyze in greater depth whether it is possible that using a more diverse portfolio of hypothetical scenarios could lead to overall improved results, and if so, whether only scenarios of sufficient likelihood need even be included in $\mathcal{H}$ to have a meaningful impact. As the runtime of the algorithm scales with the size of $\mathcal{H}$, ensuring that $|\mathcal{H}|$ is small while still covering a sufficient set of off-nominal scenarios is important for an effective mission.

Related Work

Onboard planning and execution are of great interest to the space domain. The *Remote Agent* was an architecture for onboard planning and execution addressing remote autonomous operation with deadlines, resource constraints, and concurrent activities (Muscettola et al. 1998). The Remote Agent flew for 48h in 1999 on the Deep Space One spacecraft using a batch planner that took hours on a RAD6000 CPU to generate a temporally flexible plan that was then used by a reactive executive controller (Pell et al. 1997) to provide robust plan execution. The planner used a refinement search paradigm (Jónsson et al. 2000) to construct a temporally flexible plan but did not consider utility in plan generation and did not perform continuous replanning due to the computational expense and long planning time (indeed the replans were scheduled in the prior plan).

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 limited model of utility. This scenario is similar to that proposed in this paper, in which the lander must react to dynamic events and observations in order to maximize its utility, while still adhering to both mission and spacecraft constraints. 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 flies an onboard planner (Rabideau and Benowitz 2017) to reduce lost productivity from following fixed time conservative plans (Gaines et al. 2016). The M2020 planning architecture 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, many characteristics of the M2020 mission are fundamentally different from the concept mission we consider here, such as the lack of reliable a priori model parameters, the non-repletable battery, and the long communications blackout time windows incentivizing greater mission autonomy.

Due to the presence of continuous state variables and the necessity of modeling concurrent actions, we find that stochastic planning models such as MDPs do not support our problem domain well while still being efficient to solve. Instead, as our problem has additional structure in how tasks are conditioned, we represent our model as a hierarchical task network (HTN). Hierarchical task networks have been extensively studied over the last several years as efficient models for planning in highly structured domains where expert knowledge can be embedded directly into the planner (Kuter et al. 2009; Macedo and Cardoso 2004).

Several planning algorithms have been proposed for solving HTNs (Erol, Hendler, and Nau 1994; Nau et al. 2003; Kuter et al. 2005). The subroutine that is used in Algorithm 1, `HTNSearch`, is most similar to SHOP2 (Nau et al. 2003). However, while SHOP2 selects task nondeterministically from the available task at each iteration of the planning loop, `HTNSearch` does not commit to a task but instead heuristically searches the tree of (partial) plans and deterministically selects the highest utility node found.

While the motivations and ideas of our approach are similar to the area of robust optimization and uncertainty sets, our work differs from prior work (Ben-Tal, El Ghaoui, and Nemirovski 2009; Bertsimas and Brown 2009) in two key aspects. First, robust optimization is a method for avoiding solutions to convex optimization problems that end up being infeasible in practice due to the realizations of uncertain parameters. However, as our problem is an indefinite planning problem (and replanning is not modeled as part of the planning problem), formulating it as a convex optimization problem is not straightforward. Second, uncertainty sets are often built from sampled data in the absence of well-defined priors; however, in our case, we assume the existence of well-defined priors over the uncertain parameters (e.g. battery life). We do not use these distributions during planning as full stochastic planning is intractable given the computational resources of the lander.

Conclusion

Planning and scheduling tasks in the presence of large a priori uncertainty is a challenging problem for space-based missions. The plans need to be robust and effective while not risking compromising system safety or mission success even in the face of domain uncertainty and severe computational constraints. These issues are exacerbated in the context of the Europa Lander concept mission where there is a monotonically decreasing battery supply and large windows of communication blackouts. In this work, we have presented a deterministic planning algorithm – `HTNSearch-PHRA` – that functions by creating a set of hypothesized scenarios, running an efficient HTN heuristic search planner for each scenario to produce a set of plans that are then each evaluated across all hypothesized scenarios, and returning the plan that performed the best overall. We validated the approach empirically on a simulated Europa Lander domain where we compared it against existing baselines across five different stochastic mission scenarios. We demonstrated that the approach is more robust to off-nominal deviations and unexpected scenarios than the existing baselines, having consistently better performance while still being computationally efficient (running on the order of a few seconds).

There are several topics of consideration for future work. So far, we have tested `HTNSearch-PHRA` only in the context of varying a single scenario parameter: battery supply. The natural next step is to observe how our algorithm performs when varying other parameters such as task failures, sample target data loads, communication efficiency, task utilities, and catastrophic events. Second, we provided empirical evidence that increasing the number of hypothesized scenarios, at least when only a single parameter is varied, has diminishing returns with respect to utility but grows quickly in runtime. In the future, we would like to perform a more rigorous analysis of the conditions under which increasing the set of hypothesized scenarios will be beneficial, or identify if there are conditions under which new hypothesized scenarios will not impact the results of the algorithm. Finally, we would like to perform more empirical evaluations of our algorithm on a wider set of mission scenarios where the system can perform online model updates as it makes observations that enable it to update it model priors.

Acknowledgments

This 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

Agrawal, J.; Chi, W.; Chien, S.; Rabideau, G.; Kuhn, S.; and Gaines, D. 2019. Enabling Limited Resource-Bounded Disjunction in Scheduling. In *IWPSS*.

Ben-Tal, A.; El Ghaoui, L.; and Nemirovski, A. 2009. *Robust Optimization*. Princeton University Press.

Bertsimas, D.; and Brown, D. B. 2009. Constructing Uncertainty Sets for Robust Linear Optimization. *Operations Research* .

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* .

Chi, W.; Agrawal, J.; Chien, S.; Fosse, E.; and Guduri, U. 2019. Optimizing Parameters for Uncertain Execution and Rescheduling Robustness. In *ICAPS*.

Chi, W.; Chien, S.; and Agrawal, J. 2020. Scheduling with Complex Consumptive Resources for a Planetary Rover. In *ICAPS*.

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 *ICAPS*.

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* .

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* .

Chien, S.; Tran, D.; Rabideau, G.; Schaffer, S.; Mandl, D.; and Frye, S. 2010. Timeline-based space operations scheduling with external constraints. In *ICAPS*.

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 *ICAIPS*.

Chong, E. K.; Givan, R. L.; and Chang, H. S. 2000. A framework for simulation-based network control via hindsight optimization. In *CDC*. IEEE.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. UMCP: A Sound and complete procedure for hierarchical task-network planning. In *ICAIPS*.

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 *ICAPS Workshop on Planning and Robotics*. London, UK.

Goldman, R. P.; and Kuter, U. 2019. Hierarchical Task Network Planning in Common Lisp: the case of SHOP3. In *European Lisp Symposium*. Zenodo.

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

Hansen, E. A.; and Zilberstein, S. 2001. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* .

Hoffmann, J.; and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* .

Jónsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. D. 2000. Planning in Interplanetary Space. In *ICAIPS*.

Korf, R. E. 1990. Real-time heuristic search. *Artificial intelligence* .

Kuter, U.; Nau, D.; Pistore, M.; and Traverso, P. 2009. Task decomposition on abstract states, for planning under nondeterminism. *Artificial Intelligence* .

Kuter, U.; Nau, D. S.; Pistore, M.; and Traverso, P. 2005. A Hierarchical Task-Network Planner based on Symbolic Model Checking. In *ICAPS*.

Macedo, L.; and Cardoso, A. 2004. Case-based, decision-theoretic, HTN planning. In *ECCBR*. Springer.

Muscettola, N.; Nayak, P. P.; Pell, B.; and Williams, B. C. 1998. Remote agent: To boldly go where no AI system has gone before. *Artificial intelligence* .

Nau, D. S.; Au, T. C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *JAIR* .

Pell, B.; Gat, E.; Keesing, R.; Muscettola, N.; and Smith, B. 1997. Robust periodic planning and execution for autonomous spacecraft. In *IJCAI*.

Pineda, L. E.; and Zilberstein, S. 2014. Planning under uncertainty using reduced models: Revisiting determinization. In *ICAPS*.

Rabideau, G.; and Benowitz, E. 2017. Prototyping an Onboard Scheduler for the Mars 2020 Rover. In *IWPSS*.

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 *IWPSS*.

Wang, D.; Russino, J. A.; Basich, C.; and Chien, S. 2020. Using Flexible Execution, Replanning, and Model Parameter Updates to Address Environmental Uncertainty for a Planetary Lander. In *I-SAIRAS*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *ICAPS*.

Yoon, S. W.; Fern, A.; Givan, R.; and Kambhampati, S. 2008. Probabilistic Planning via Determinization in Hindsight. In *AAAI*.

Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI magazine* .