Using Flexible Execution, Replanning, and Model Parameter Updates to Address Environmental Uncertainty 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
firstname.lastname@jpl.nasa.gov

Abstract

Planning for unknown environments presents a number of technical challenges. The planner must ensure robustness to unknown phenomena and manage unpredictable variation in execution, all while operating in a capacity that maximizes its objective. Productivity in the face of these challenges requires an integrated approach to planning and execution that is capable of accomplishing goals, reacting to variation, and maximizing overall utility. We examine this problem in the context of a Europa Lander concept mission. We model the problem as a hierarchical task network, framing it as a utility maximization problem constrained on a depletable energy resource. We propose a planning and execution framework that responds to feedback using three techniques: (1) flexible execution, (2) periodic replanning, and (3) online model parameter and utility updates. The efficacy of each of these techniques is examined through simulation of a Europa Lander concept mission, showing higher utility achievement compared to baseline approaches. We demonstrate that an integrated approach to planning and execution that is grounded in replanning, utility maximization, and model parameter updates 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. We use a hierarchical task network (HTN) that defines activities and their associated dependencies, and we create plans using this mission representation. To drive planning goals, we assign utility to tasks that complete those goals, and generate a plan that maximizes utility while obeying each task's constraints. At execution time, we use flexible execution and re-planning to react to uncertainty and variation, and to compensate for an approximate world model. Finally, we accept model parameter updates (such as expected task energy usage and estimated task utility) during execution time to incorporate execution feedback and gained knowledge and use these updated models to optimize plan quality.

We examine the problem in the context of a proposed 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 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.

We integrate planning and execution in order to better react to environmental variation, moving away from fixed time and energy budgets to generate less conservative, more successful plans. To do so, we use MEXEC, an integrated planner and executive first built for NASA's Europa Clipper mission (Verma et al. 2017). To account for approximate world models, our framework replans on a periodic basis to recalibrate its plans with reality. Finally, we integrate a module that continually estimates task parameters and goal utilities during execution in response to new information. We update

[1] 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.

the priors on the relevant model parameters accordingly, and subsequently replan to produce a more accurate plan. This further improves plan quality and utility gain.

We present this planning and execution framework in a simulated Europa-like mission and compare it to three baseline approaches similar to those used in prior missions: a static plan (Gaines et al. 2016a), flexible execution with no replanning, and flexible execution with replanning but without online parameter update (Rabideau and Benowitz 2017). We explore the value of flexible execution, replanning, and online model reparameterization, and examine their effect on utility in these scenarios. We present empirical results of the efficacy of each technique over the others, and discuss further techniques and implementations that may continue to improve on these baselines into the future.

Problem 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 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. In addition, the lander may choose to resample the same location, for example, in order to verify the discovery of a biosignature at that location. 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. Because communication is difficult and energy intensive, the lander may choose to compress data lossily 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 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

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

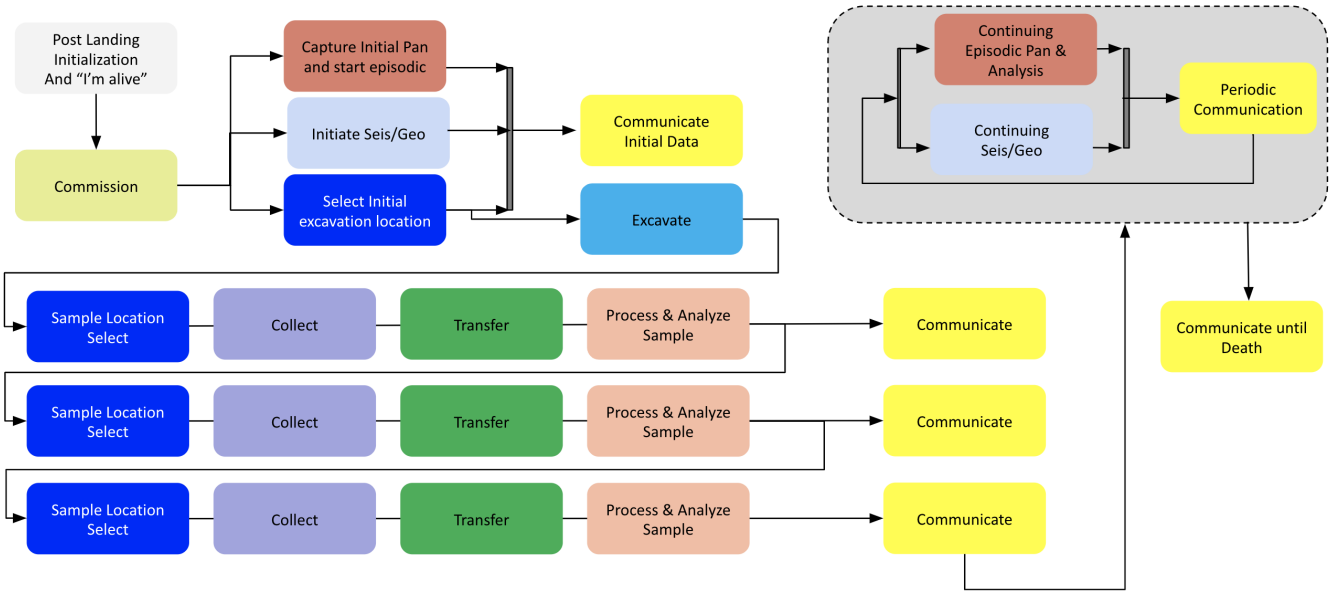

Figure 1: A task network for the Europa Lander mission concept. The diagram represents a potential execution trace of the mission that would fulfill baseline requirements.

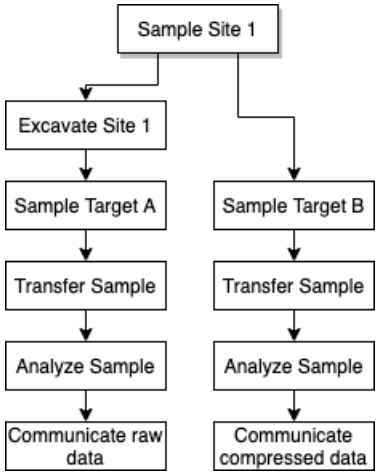

Figure 2: Two possible decompositions of a single parent "Sample Site 1". In the left decomposition, the lander excavates the site, samples target A, and communicates raw data. In the right decomposition, the lander skips excavation, samples site B, and communicates compressed data. Both achieve the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

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.

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. The planning algorithm is described below, and provided in pseudocode in Algorithm 1.

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 cost of each task scheduled in the node's 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.

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;
        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;
```

Execution

Planning and execution are integrated in our approach, in order to respond to variation and therefore better optimize overall utility achieved. We use MEXEC for flexible execution, which allows us to make small plan modifications to task start times and resource consumption without failing (Verma et al. 2017). We maintain task models that have impacts on certain resources, e.g. each task has some negative impact on an energy resource, representing its energy usage. As is common in robotics, variation during execution time

may prove these models to be incorrect. Thus, during execution, we update our resource timelines to match the values measured.

In addition to updating the raw value of modeled resources, we also update task impact models with the information gained during execution. This is especially important in the sampling tasks. There exists significant uncertainty in our model of interactions with the Europan surface, so we may discover that collecting a sample is much more difficult than previously expected. In this case, we rely on updating our model of the sample collection task to better represent what has been discovered. By doing so, we are able to replan with a better understanding of the task, thereby creating plans that are more likely to succeed. In this paper, we explore only one online parameter update policy, where the latest energy use value is used to update the task model, i.e. our prior estimate is ignored after gaining some posterior knowledge. Future work would explore more sophisticated policies and their resulting effects on utility achievement.

We also update our utility estimation for certain tasks. Our task network has a prior estimate on the utility of collecting surface samples that is uniform across all targets. At execution time, the information discovered may drastically alter this estimation. For example, after sample analysis, the lander may discover a biosignature at target A, but nothing at target B. Our framework would then update its task models accordingly, rewarding sample collection tasks at target A with much higher utility than tasks at target B. Online utility update thus allows us to redefine our goals and the value of each goal according to the information obtained at execution time. We again restrict our focus to a single policy of online utility update, using the latest value to overwrite any prior estimate.

Tying all of this together is periodic replanning on a fixed cadence. This allows us make use of online state updates, model updates, and utility reassignments during execution time, thereby creating plans that achieve higher overall utility. 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 us to update our goals according to what is realistically possible given the current state measurements of the system. In future work, we may replan in response to events such as the detection of significant error in resource modeling (Chi et al. 2018). This would minimize replanning, such that it would occur only when necessary.

Results

We examine the performance of four potential planning and execution frameworks in a simulated Europa Lander problem. As a baseline, we examine the performance of a static plan, executed with fixed time points and energy impacts. We refer to this strategy as "none", as it uses none of the flexible execution techniques we describe. While this strategy is extremely basic, it is also the baseline for many NASA missions, including the Curiosity rover (Gaines et al. 2016a)(Gaines et al. 2016b). With this strategy, we generate a static plan and attempt to execute it, without any flexibility. Because we lack flexibility, the static plan must contain margin such that tasks are unlikely to overrun their allotted resource budgets. If any overruns do occur, we terminate the execution of the plan.

We refer to the second strategy as "flexible". With this strategy, we allow flexible execution of a plan using MEXEC, but do not replan during execution. Because execution is flexible, we can reduce the amount of margin in our task models, and rely on our executive to handle resource overruns accordingly. In a real application, this would be analogous to placing an executive on a spacecraft, but only allowing it to run a plan generated on the ground.

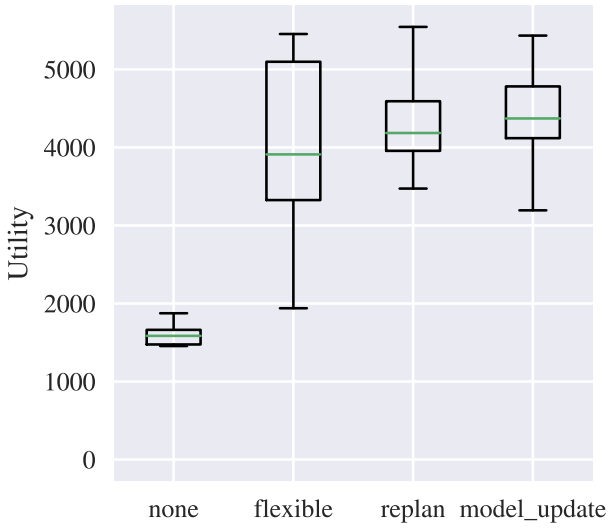

Figure 3: Final utility achieved by four different planning and execution strategies applied to simulated Europa Lander scenarios.

The third strategy is "replan", where we allow for replanning during execution, but do not perform any model parameter or utility updates. At a periodic cadence, we run the planning algorithm using updated resource values, but we do not incorporate any information gained about task model parameters or utility. This baseline is comparable to expected operations on the Mars 2020 mission (Rabideau and Benowitz 2017).

Finally, the "model_update" strategy incorporates all techniques described in the approach, executing plans flexibly, replanning on a regular cadence, and updating task model parameters and goal utilities online. Specifically, we estimate and update parameters for the sample collection task. This task is particularly important because it uses significant energy and is repeated many times in the plan. In addition, as a task that interacts with the Europan surface, our model of the task is likely to be extremely uncertain.

To evaluate the efficacy of these approaches, we simulate execution on a Europa Lander problem. In this simulation, we vary the energy usage of each task, drawing from a normal distribution centered around the prior estimate in the

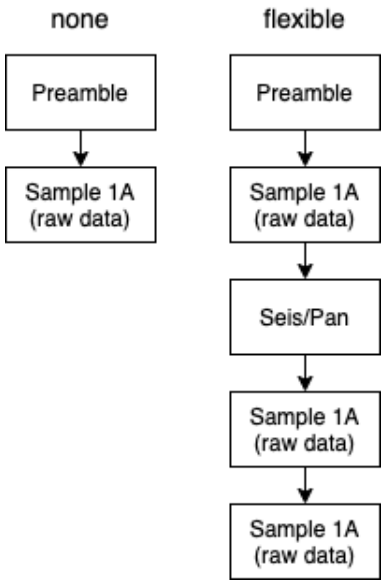

Figure 4: Sample high-level plans for the none and flexible planning/execution strategies.

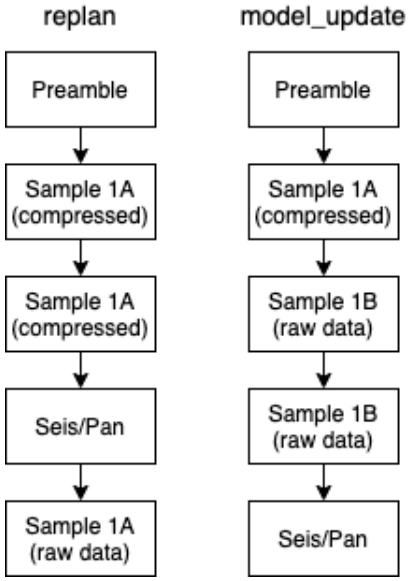

Figure 5: Sample high-level plans for the replan and model_update planning/execution strategies.

input task network. In addition to this variation in energy, we vary the utility of different sampling targets. We draw this value from a high-variance normal distribution, again centered around the prior estimate in the task network. Finally, to simulate uncertainty in our model, for the sample collection task, we draw energy use from a normal distribution centered around a mean that does not match the task network input. Instead, the mean is drawn from another normal distribution, centered around the input. This means that the energy distribution expected in the world model may not match the true distribution.

The results of our experiment are shown in Figure 3. Each strategy was simulated for 50 runs, recording the average utility achieved by the execution trace. We find that the "none" strategy's static plan is severely limited by the margin required to ensure that the plan can be executed without resource overruns. This margin is two standard deviations above mean, and limits our plan to only sampling a single time. Even with this margin, we find that resource overruns still occur, which result in no utility achieved. The static approach achieves a median utility of 1585 and mean of 1325.

Allowing flexible execution with the "flexible" strategy improves this significantly, bumping median utility gain to 3911, and mean utility to 4195. This benefit comes primarily in the removal of the margins. With an onboard executive, our plans can expand to the full set of 3 samples, which greatly increases the ceiling for our utility gain. Figure 4 contrasts the plans of the "none" and "flexible" strategies at a high level. These plans were selected from execution traces of the experiment. While flexible execution is very valuable, because we do not replan, we find that the utility achieved by this strategy varies widely. Plans are fairly brittle to unexpected energy loss, often resulting in missing the final communication step in the plan. This proves disastrous for utility gain, because the work required to generate the final data product is essentially wasted. Thus, we see a wide spread of overall utility in this strategy, with two main clusters: one for traces where all three samples are successful, and one for traces where one sample fails.

With the "replan" strategy, we allow replanning, which significantly reduces the brittleness of our plans to variation. We find that this reduces the variance of utility achieved, but does not significantly affect the overall utility gained (median 4185, mean 4027). Figure 5 shows a sample plan with this strategy. Because we react to variation in energy, the planning algorithm begins to favor compressed data as a way to save on energy while still achieving sampling goals. However, this also cuts down on potential utility. Because the flexible strategy is so optimistic, it is bimodal in utility gain, either achieving close to optimal, or falling far short. The replan strategy tends toward a middle path, with a lower variance of moderate utility plans. This variance reduction is useful for the Europa Lander problem, since the cost of failure is significant.

Finally, the "model_update" strategy results in a median utility gain of 4372 and mean of 4327. The most significant difference in the full strategy is the online re-weighting and re-parameterization of different sampling tasks. Depending on what is discovered at a given sample target, we update the utility of the sampling action corresponding to that target. This motivates the lander to sample in other locations if a particular target is found to be less valuable or more costly than expected. Because the planner can take advantage of this newfound knowledge, it is able to harvest greater utility from sample targets. In our sample plan, we find that this case occurs in practice: After discovering that sampling target A is less interesting or more difficult than originally predicted, the lander chooses to swap targets to target B. It is thus able to make better use of energy and more effectively

harvest utility, compared to methods that do not estimate parameters online.

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 statesapce.

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. It has been shown that solving an MDP completely is P-complete in the size of the statespace, and solving a POMDP PSPACE-complete, but the size of the state space in these models is itself exponential in the variables that represent it (Littman, Dean, and Kaelbling 1995; Papadimitriou and Tsitsiklis 1987). This computational burden 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 requiring frequent replanning. Additionally, the critical nature of these missions frequently mandates a level of interpretability for human operators in the schedules or plans generated for validation and simulation purposes that is not always afforded by the complex processes used by the solvers to produce their solution.

Search-based algorithms have been a popular alternative to these approaches 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 our case, all three of these properties are highly desirable, and influenced our decision to utilize a heuristic search-based planning algorithm. Additionally, our problem has additional structure in how tasks are conditioned allowing us to represent our input as a heierarchical task network (HTN). HTNs have been extensively studied over the last several decades as efficient algorithms for planning in highly structured domains where expert knowledge can be embedded directly into the planner (Erol, Hendler, and Nau 1994; Kuter et al. 2009; Macedo and Cardoso 2004). Our formulation is most similar to that of SHOP2 (Nau et al. 2003), a forward chaining planning algorithm with search-control heuristics and deterministic outcomes. However, unlike in SHOP2 where at each iteration of the loop the next task is selected nondeterminstically from the available tasks, our algorithm does not commit to a task, but rather heuristically explores the search tree of partial plans, where each node is a partial solution to the HTN, and then deterministically selects the plan that has the highest utility. Similar is the idea of combining Monte-Carlo tree search (MCTS) with HTNs (Ontanón and Buro 2015). However, this work focuses on improving the efficiency of MCTS in adversarial domains with enough structure for an HTN representation to be effective.

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 plans to fly an on-

board planner (Rabideau and Benowitz 2017) to reduce lost productivity from following fixed time conservative plans (Gaines et al. 2016a). 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, 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

In our planning, we consider only one dimension of decision theory: utility. While we react to uncertainty at execution time, we do not take this into account during the planning phase. A more sophisticated planner would explicitly integrate probability into plan generation, maximizing expected utility rather than assuming resource impacts are constant and correct. 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.

In addition, considering depletable resource usage probabilistically would allow optimization of plans to avoid significant utility loss if that resource is depleted. For Europa Lander, energy use is a significant constraint on all tasks. By reasoning about the probability of running out of energy, we can improve our plans and avoid these cases when possible. One simple approach would be to discount a task's utility according to the energy remaining at the time it is scheduled. This is analogous to the intuitive approach of discounting a task's utility according to its distance into the future. Instead of tying utility discounts to time, however, we tie it to energy, because this is the most significant task constraint in the Europa Lander problem.

A more sophisticated approach would examine plan prefixes and choose the one that best balances potential utility and risk. Here, we would generate a set of plan prefixes that plan until some fixed time or energy point is reached, e.g. the start of the next planning cycle. Given this plan prefix, we would generate plan postfixes given various energy levels, drawn from a distribution representing the battery remaining. Using a sampling approach, we could arbitrarily refine our estimate of the plan prefix's quality given various energy levels. Then, we would pick the plan prefix that maximizes the overall expected utility of the entire plan. Adding probabilistic reasoning into our planning algorithm is likely to improve plan quality and thus improve the overall performance of our planning and execution framework.

Acknowledgments

Pre-decisional information – for planning and discussion purposes only. 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. Also appears at the 29th International Conference on Automated Planning and Scheduling (ICAPS) Workshop PlanRob 2019 and ICAPS SPARK 2019 and ICAPS IntEx 2019.

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

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence (AIJ)* 129(1-2):5–33.

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)*. Also appears at International Symposium on Artificial Intelligence, Robotics, and Automation for Space (ISAIRAS 2018) as an abstract.

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 in-

telligent payload experiment cubesat mission. *Journal of Aerospace Information Systems* 14(6):307–315.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. UMCP: A Sound and complete procedure for hierarchical task-network planning. In *International Conference on Artificial Intelligence Planning Systems (AIPS)*, volume 94, 249–254.

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

Gaines, D.; Doran, G.; Justice, H.; Rabideau, G.; Schaffer, S.; Verma, V.; Wagstaff, K.; Vasavada, A.; Huffman, W.; Anderson, R.; et al. 2016b. Productivity challenges for Mars rover operations: A case study of Mars Science Laboratory operations. Technical report, Technical Report D-97908, Jet Propulsion Laboratory.

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.

Hansen, E. A., and Zilberstein, S. 2001. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence (AIJ)* 129(1-2):35–62.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14:253–302.

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)*.

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

Korf, R. E. 1990. Real-time heuristic search. *Artificial intelligence (AIJ)* 42(2-3):189–211.

Kuter, U.; Nau, D.; Pistore, M.; and Traverso, P. 2009. Task decomposition on abstract states, for planning under nondeterminism. *Artificial Intelligence (AIJ)* 173(5-6):669–695.

Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 394–402.

Macedo, L., and Cardoso, A. 2004. Case-based, decision-theoretic, HTN planning. In *European Conference on Case-Based Reasoning*, 257–271. 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 (AIJ)* 103(1-2):5–47.

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.

Ontanón, S., and Buro, M. 2015. Adversarial hierarchical-task network planning for complex real-time games. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Pell, B.; Gat, E.; Keesing, R.; Muscettola, N.; and Smith, B. 1997. Robust periodic planning and execution for autonomous spacecraft. In *IJCAI*, 1234–1239.

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.; 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.

Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI magazine* 17(3):73–73.