# Runtime Decision Sampling and Branch Utility Re-Evaluation in the Robust Execution of Contingent Plans

Emmanuel Benazera
RIACS, NASA ARC, Moffet Field, CA
ebenazer@email.arc.nasa.gov

Richard Levinson, Thomas K. Willeke
QSS Group Inc, NASA ARC, Moffet Field, CA
rich,twilleke@email.arc.nasa.gov

Howard H. Cannon
NASA ARC, Moffet Field, CA
howard.h.cannon@nasa.gov

***Abstract*— Current planning algorithms have difficulty handling the complexity that is due to an increase in domain uncertainty, and especially in the case of multi-dimensional continuous spaces. Therefore, they produce plans that do not take into account numerous situations, such as faults or other changes in the planning domain itself and that can occur at runtime. Here, we present our approach to the robust execution of contingent plans, i.e. that involve conditional branches based on decision functions of the system state. We use a version of the Monte-Carlo simulation for Markov Decision Processes to re-evaluate branch utility values and branch conditions whenever an unpredictable event occur, based on health monitoring information at runtime.**

***Index Terms*— Execution under uncertainty, Contingent planning, Monte Carlo simulation.**

## I. Introduction

The need for autonomy and robustness in the face of uncertainty is growing as planetary rovers become more capable and as missions explore more distant planets. Recent progress in areas such as instrument placement [1] makes it possible to visit multiple rocks in a single communication cycle. This requires reasoning over much longer time frames, in more uncertain environments. Simple unconditional plans as used by the Mars Exploration Rovers (MER) will probably have a very low probability of success in such context, so that the robot would spend almost all its time waiting for new orders from mission control.

Recent architectures for future planetary rover missions include a planner/scheduler, a health monitoring system, and an executive. The planner/scheduler generates a control program/plan that describes the sequence of run-time actions necessary to achieve mission goals. Since the rover's environment is highly uncertain [2], the control programs (also called *plans*) are contingent plans [3] in that they involve conditional branches that are based on decision functions of the system state that the executive can evaluate in real time. The executive is responsible for the execution of the control programs, taking into account the current state of the system as estimated by the health monitoring system. This capability, also referred to as *robust execution under uncertainty*, includes deciding the best branch in a plan when reaching a branch point, given an estimate of the current system state, inserting and replacing plan portions to react to faults and other unpredictable events.

Typically, planning proceeds to a mapping from the system state space to the utility space, i.e. the utility obtained by executing the plan, that it seeks to maximize. Noting the system state $s = (x, r)$ with $x \in X$ the discrete state (or system modes), and $r \in R$ the multi-dimensional continuous state (including time), the utility earned by executing a branch $b_i$ starting at $s$ can be noted:

$$V_{b_i}(s) = \sum_{x' \in X} \int_R p((x', r') \mid s, a_{i1})[U(a_{i1}, (x', r')) + V_{B_i}(x', r')]dr' \quad (1)$$

with $a_{i1}$ the first action of branch $b_i$, $B_i$ the remaining portion of the branch, $U(a_{i1}, (x', r'))$ the utility earned, and $s'$ the system state after executing $a_{i1}$ following the probability distribution $p(s' \mid s, a_{i1})$. Over a belief state $b(s)$, as estimated by the health monitoring system, we have:

$$V_{b_i}(b(s)) = \sum_{x \in X} \int_R V_{b_i}(x, r) b(x, r) dr \quad (2)$$

And at a branch point where $n$ branches are available, the best branch is decided according to:

$$b^* = arg \max_{i \in [1,n]} V_{b_i}(b(s)) \quad (3)$$

This is similar to the Bellman equations for the Partially Observed Markov Decision Processes (POMDPs)[4], [5]. But such planners have difficulties handling certain situations, such as actions that carry no utility (typically used for responding to unlikely situations), fault occurences and belief state update. Actions with no reward can usually be inserted anywhere in the plan at low cost, so the greedy approach that seeks to maximize the expected utility (relation (3)) fails to position them efficiently. Second, most planners use a nominal model of the world and system actions, not representing potential faults (e.g. stuck wheels, broken navigation system, rocky environment, etc...). Moreover, fault models exponentially increase the complexity of the planning even if the faults have low probability of occurence as they can occur at any time. Finally, the health monitoring system returns an ever changing belief state over time that has to be taken into account (relation (2)). For these reasons, the response to unlikely situations and faults is better decided at execution: the health monitoring system passes a belief over system state to the executive

that decides which portion of the plan to execute, sometimes inserting/replacing wanted/unwanted plan blocks.

In general, the problem can be seen as one of re-evaluating online the utility value of branches of the contingent plan to decide the best of these branches. In this paper we show how to adapt the classical technique of Monte Carlo (MC) simulation [6], [7] to the re-evaluation of contingent plans and decision under uncertainty. The novelty lies in an algorithm that makes near optimal decision and computes branch decision lines through piecewise constant approximations of value functions, based on Bellman equations. The integrated approach is to be tested on a Mars-type rover simulator this month, and is the first step toward the development of new techniques for both planning and execution under uncertainty in the coming years. We start by giving the reader a brief description of a contingent plan and the need for plan re-evaluation.

## II. Contingent plans, and floating contingencies

The work that is presented in this paper builds on top of a larger effort, the development of theory and tools for planning and execution under uncertainty [8], [9]. Briefly, a contingent plan can be seen as a tree of branches, that are chains of actions with uncertain cost in term of a set of continuous resources, typically energy and time. A temporal network represents and propagates the time constraints among the actions and branches. An addition to this framework is one of having floating contingencies, i.e. branches that can be plugged in the plan tree at any time, depending on conditions over the resource state. These branches are typically used to react to unpredictable situations. The nodes of the tree are known as the branch points of the contingent plan. The value function for a node is a continuous function over the multi-dimensional resource state, i.e. a mapping from the resource space to the utility space, and depends on downstream node value functions. Planning determines the conditions over the resource space that discriminate among the branches at a given branch point [3]. Thus when the plan execution reaches a branch point, the branch conditions are evaluated to determine which branch is to be executed next. But due to uncertainty on the resource state or because the conditions and hypothesis under which the plan has been created have changed, the branch conditions can become nonvalid and must be re-evaluated. Note that whenever these changes occur, the mission objectives however remain the same, so the topology of the contingent plan remains unchanged but for the insertion/replacement of portions of the plan to respond to unpredictable situations.

### A. A contingent plan for the Mars exploration domain

Consider the plan for a Mars rover on figure 1. It tells the rover to first navigate to a waypoint $w_0$, and there to decide whether to take a high resolution image of the point (HI res) or to move forward to a second waypoint $w_1$ depending on the level of resources (here energy and time). After reaching $w_1$ and digging in the soil, it must decide whether to move forward to waypoints $w_3$ or $w_2$ or to simply get an image of $w_1$ and wait for further instructions. NIR is a spectral image of a site or rock. Action time and energy consumptions are represented as Gaussian bumps over empirical mean and variance.

### B. Computation of branch conditions

To determine the branch conditions, planning needs to approximate the branch value functions at branch points. Each function maps the resource space to the utility of the branch. The $\max$ operator of relation (3) defines an upper bound on the branch point overall utility value, and branch conditions are found at the functions intersections. Now consider figures 1(b) and 1(c), where branch value functions are pictured for both branch points $bpt_1$ and $bpt_2$. Intersections define branch conditions $\alpha_1$, and $\alpha_2$, $\beta_2$: for example at $bpt_2$, if the measured resource (here the energy) is below $\alpha_2$, the rover will decide to execute branch $b_5$.

Unfortunately, all situations cannot be efficiently predicted at planning time, so produced functions are not valid in all cases. For example, action costs can change dramatically during execution, due to the possible occurence of faults (e.g. a faulty wheel that slows rover's navigation and changes the time/energy cost of the action). Similarly, plan portions insertion/replacement approximations that have been developed [10] fail here as they rely on pre-computed value functions.

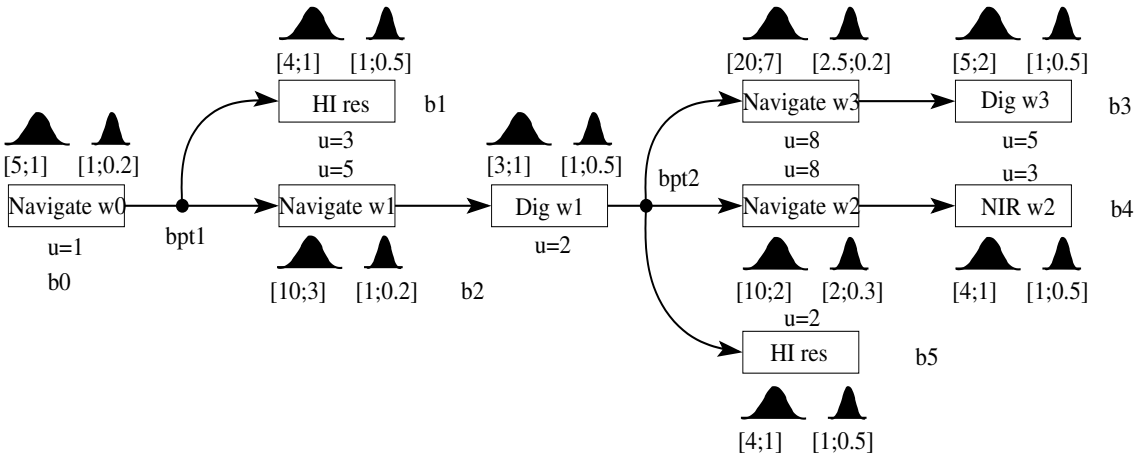
These changes produce a shift in the value functions, thus perturbating the functions intersection, and modifying the branch conditions. Moreover, a shift at one branch point implies shifts on value functions of all other branch points downstream in the plan. Therefore, when changes occur, it becomes necessary to re-evaluate large portions of the plan.

### C. When to re-evaluate the contingent plan

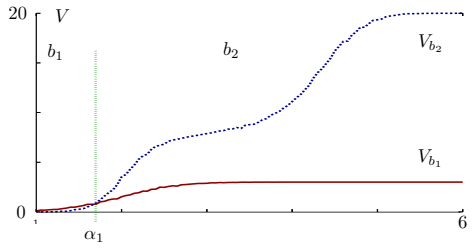
There are several conditions and situations under which the plan value must be re-evaluated. First, when the execution encounters a branch point, any change in the Bellman equation functions, such as the belief $b$ over the state $s$, the reward model $U$, the action cost model, requires that all branch functions at this branch point are re-evaluated. Second, if not at a branch point, but if a floating branch has to be inserted, then the plan equation is changed and the remaining portion of the main branch as well all future branch conditions must be re-evaluated. For example, when inserting a branch $b_f$, equation (1) becomes:

$$V_{b_f}(s) = V_{b_f}(s) + \sum_{x' \in X} \int_R p((x', r') \mid s, b_f) V_B(x', r') dr' \tag{4}$$

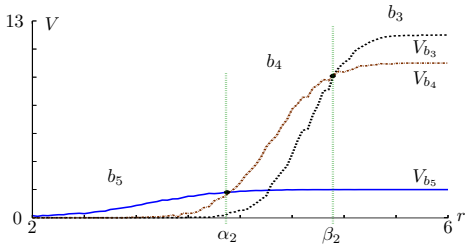
where $B$ is the remaining portion of the current plan to be executed after $b_f$. The local value of $b_f$ is the expected reward from the actions within the floating branch itself. The remaining term is a representation of the end state of the local plan, including the probability of the resources remaining after executing the local plan.




(a) Contingent plan for the Mars rover domain




(b) Value functions of branches at branch point 1 (bpt1)




(c) Value functions of branches at branch point 2 (bpt2)

Fig. 1. Branch value functions at branch point for a detailed rover problem

For all these reasons, plus the difficulty of computing the recursive multi-dimensional integrals of equation (1), we adopt an MC approach as a generic technique capable of solving all these problems together at execution time by re-evaluating the value functions, whose new intersections are later approximated. Another solution is a discretized Dynamic Programming backup. However it is less flexible than the MC approach and requires approximations of the planning domain to become tractable [9].

## III. The Monte-Carlo approach to the online re-evaluation of contingent plans

### A. Approximating branch average utility

Applying Monte Carlo techniques to the approximation of equation (2) is straightforward: the integral over the multi-dimensional continuous space is turned into a sum by sampling $N$ times from $b(s)$ and $p(s' \mid s, a)$, and the utility is averaged over the successive runs. We note:

$$\hat{V}_{b_i}(b(s)) = \sum_{x \in X} \sum_{x' \in X} [U(a_{i1}, s'_j) + \hat{V}_{B_i}(s'_j)] \quad (5)$$

$$\hat{V}^{std}_{b_i}(b(s)) = \frac{1}{N} \sum_{j=1}^{N} \hat{V}_{b_i}(b(s)) \quad (6)$$

where $s'_j \sim p(s' \mid s_j, a_{i1})$ and $s_j \sim b(s)$. The larger the $N$, the better the fit to the underlying probability distributions, and the better the approximation. Note that the major drawback of the Monte-Carlo approach is that it provides a probabilistic guarantee of its results, that is never absolute. In this work, given the complexity of the studied planning domain, we assumed we would always dispose of a sufficient number of samples.

### B. Plan simulation

For simulating branches with MC, we use a prioretized pile of events including plan actions, and a set of constraints among them. The pile is filled up with actions whose execution is simulated by testing their temporal constraints and sampling their costs before being rewarded and popped out. Floating branches are a challenge to the simulator because they can trigger at anytime. The simulator uses random events to trigger these branches and special dynamic constraints to handle their insertion. The complexity increase due to floating branches is a product of the number of plan actions, actions in the branch, and the number of these branches.

### C. The simple strategy to the maximization of earned utility

The simplest algorithm for approximating equation (3) relies on the observation that typical plans for the exploration rover contain a few branch points, with a couple of branches per point. The algorithm builds all $K$ paths $p_k$

in the contingent plan, then approximates each path utility with:

$$\hat{V}_{p^k} = \sum_{b_i \in p^k} \hat{V}^{std}_{b_i}(b(s)) \quad (7)$$

and selects the best path in the plan:

$$b^* = arg \max_{k=[1,K]} \hat{V}_{p^k}(b(s)) \quad (8)$$

Algorithm 1 returns the best path over the entire resource

1: Generate all branch paths in the contingent plan.
2: **for all** generated paths **do**
3: Proceed with MC on the path.
4: Return the path with the highest averaged utility.

**Algorithm 1:** Path maximization of average utility

space but that is not optimal for every value in this space: certain level of resources a higher utility could be obtained by executing a different path.

### D. Sampling decisions

A solution to this problem is to sample the decision itself by deciding the path with highest utility for each sample, instead of using the $\max$ operator over a set of pre-compiled paths. We write:

$$\hat{V}^{dec}(b(s)) = \frac{1}{N} \sum_{j=1}^{N} \max_{i \in [1,n]} \hat{V}_{b_i}(b(s)) \quad (9)$$

Algorithm 2 differs from the previous one in that each

1: **for all** $j < N$ **do**
2: Proceed with MC on the first branch.
3: **for all** branches $b_i$ at branch point **do**
4: Apply this algorithm recursively to $b_i$, with $j = 1$.
5: Return the highest utility at this branch point ($\max$).
6: Return the averaged utility of the plan.

**Algorithm 2:** Recursive procedure for sampling decisions

path is explored by each sample for the evaluation of the $\max$ operator. The averaged returned utility is now near optimal, but the sampled decision for the best branch (the $\arg$ operator) depends on the sampled resource space that must be partitioned into subregions of identical decision. The next section covers the retrieval of the decision lines in the multi-dimensional resource space.

## IV. Bounding the resource space for deciding future branches

Decision at branch points can be made based on the simulation results by executing the branch with the highest earned utility average. MC simulation is not particularly efficient and the rover is forced to wait at branch points while the plan is re-evaluated. However, simulation provides sufficient information for computing branch conditions at future branch points. This operation is performed at virtually no cost and can spare future simulation by constraining future decisions while at current branch point or floating branch insertion.

### A. Producing bounds based on sampling decisions

Our first solution relies on the probabilistic guarantee of the MC approach: we seek to ensure a convex[1] resource domain $D(b_i) \in R$ for each branch $b_i$ such that all samples in $D(b_i)$ have $b_i$ as the branch with highest utility. We enlarge bounds based on each sampled decision, then strip out intersection among the branch bounds. This strategy is easy to implement but leaves undecided zones at branch points (the stripped areas), for which the MC simulation has to be re-run whenever execution reaches that point. A solution to this problem is to fit the branch sampled value functions at branch point and study their intersection to come up with a better approximation of the optimal decision lines.

### B. Approximating branch decision lines thru piecewise constant value function approximation

Our second solution is to slice the resource domain into rectangular bins and to fit the branch value functions in each bin with a piecewise constant function, based on the MC samples. Function intersections are found at bin edges. Noting $\Delta_r$ a bin in the resource space, we can write $b_i$'s value:

$$\hat{V}_{b_i}(b(s)) = \sum_{\Delta_r} \sum_{x \in X} p(b_i \mid \Delta_r) p(\Delta_r) \hat{V}_{b_i}(\Delta_r, x) \quad (10)$$

i.e. as the sum of the average utilities of $b_i$ in each bin when it is the branch with the highest expected utility. More precisely:

$$\hat{V}_{b_i}(\Delta_r, x) = \frac{1}{n_{r_{\Delta_r}}} \sum_{r_j \in \Delta_r} \sum_{x' \in X} \hat{V}_{b_i}(s_j) \quad (11)$$

with $s = (x, r)$ and $s_j = (x', r_j)$, is the average utility of $b_i$ on bin $\Delta_r$ from the $n_{r_{\Delta_r}}$ samples $r^j$ it contains,

$$p(b_i \mid \Delta_r) = \frac{1}{n_{r_{\Delta_r}}} \sum_{r_j \in \Delta_r} \delta(b_i = arg \max_{i \in [1,n]} \hat{V}_{b_i}(b(s_j)) \quad (12)$$

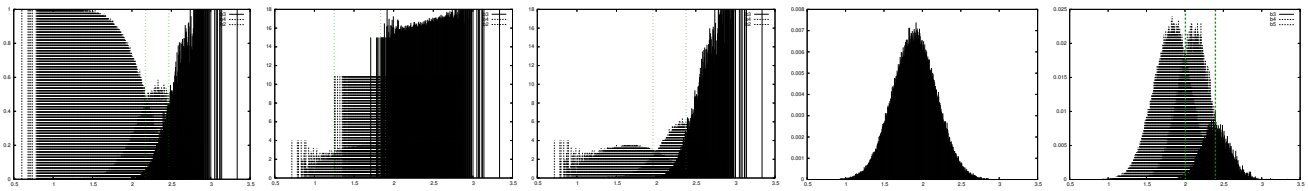
where $\delta$ is the Dirac function, is the probability for $b_i$ to be the branch with the highest utility over the samples of the bin,

$$p(\Delta_r) = \frac{n_{r_{\Delta_r}}}{N} \quad (13)$$

is the probability of the bin itself. An optimal bin size $W$ is obtained, in the sense that it provides the most efficient unbiased estimation of the probability distribution function formed by the samples. We used $W = 3.49\sigma N^{-1/3}$ where $\sigma$ is the standard deviation of the distribution, here estimated from the samples [11], [12]. The overall strategy is presented on algorithm 3.

Figure 2 pictures results for the second branch point of our rover problem (the energy is pictured and the time line is omitted) and shows the shifting branch conditions on the horizontal axis that is the energy line. Branch conditions are obtained by comparing the branch with the highest utility

[1] In the rover domain, value functions are monotonically increasing w.r.t. time and energy.



(a) $p(b_i \mid \Delta_r)$ (b) $\hat{V}_{b_i}(\Delta_r)$ (c) $p(b_i|\Delta_r)\hat{V}_{b_i}(\Delta_r)$ (d) $p(\Delta_r)$ (e) $\hat{V}_{b_i}(b(s))$

Fig. 2. Piecewise constant approximation of branch value functions from simulation samples

1: Proceed with algorithm 2 and collect samples at branch point.
2: **for all** branch points in the contingent plan **do**
3: Compute the optimal bin size and slice the space into bins.
4: Compute statistics with equations (11), (12) and (13).
5: Evaluate equation (10) for each branch.
6: In each bin, identify the branch with the highest value.
7: Identify new branch conditions where successive bins have different highest utility branches.

**Algorithm 3:** Branch conditions approximation thru piecewise constant value function approximation

for each bin: if two successive bins return different results, a branch condition exists at their edge. Thus, the precision of the approximation is directly dependent on the optimal bin size, that depends on the number of samples.

### C. Belief update on re-evaluated branch conditions

As explained in subsection II-C, they are several situations when a plan must be re-evaluated. The change in the belief state as returned by the health monitoring system is problematic because it happens at a high frequency. We observe that if no change occurs but for the belief updates, then the branch value function can be approximated based on the new belief state $b'(s)$:

$$\hat{V}_{b_i}(b'(s)) \approx \hat{V}_{b_i}(b(s)) \sum_{x \in X} \int_{D(b_i)} b'(x,r)dr \qquad (14)$$

whose last term is easily computed or estimated, and $V_{b_i}(b(s))$ is known.

## V. Results

### A. IS architecture for the execution of contingent plans under uncertainty

The Intelligent System (IS) architecture for rover autonomy at NASA Ames includes a high level contingent planner [3], a hybrid model-based particle filter as the health monitoring system [13], [14] and a concurrent executive. The planner operates offline while the particle filter and the executive run concurrently on the rover. The planner uses a mission domain and nominal rover model and produces a concurrent plan with branch points at critical time and energy points. The executive is fed with the plan and follows branch conditions at branch point until it detects a plan re-evaluation is necessary. The particle filter gets raw data from the rover sensors and estimates a hybrid state of the system and its environment: discrete wheel states (running, stopped, stucked), discrete terrain states (rocky, flat), sensor fault modes as well as numerous continuous variables to support them. For now, the executive incorporates the discrete modes only. We use pre-estimated action models for each of the faults. In the future we will look into the estimation of these action models at runtime. Our testbed has been a high-end simulator (Mars Simulation Facility, MSF) that connects to both the executive and the monitoring system.

### B. algorithms testing

*1) Monte Carlo results:* We first assess the results of both MC algorithms. We use two plans generated by the contingent planner. $MSL\ demo$ is a plan designed for the future Mars Science Lab mission rover. This is a test plan, so it has a small number of actions, and a loose temporal network. The $K9\ demo$ plan is a real plan to be executed on the rover during the upcoming demo at Ames: it contains a single branch point, but each branch has a high number of actions and a dense temporal network among them. Both algorithms run in the same time frame on the $MSL$ plan but the *Simple* decision making returns less optimal results. On the more complicated $K9$ plan, the simple algorithm takes longer to execute as it re-evaluates overlapping parts of the plan paths several times.

*2) Branch conditions results:* Second, we assess the re-evaluated branch conditions at plan branch points: bounds/bins are generated with the sampling decision algorithm, and verified by running a classical Monte-Carlo simulation, that does not maximize the utility, but follows the new branch conditions and averages the earned utility. Simulation also returns the failure probability of the plan. The error is the difference to the optimal plan value in percentage. The piecewise constant approximation of the branch value functions returns highest utility and lower failure probability.

### C. Running example

Our running example has the rover exploring three rocks on MSF and performing twenty eight actions including

| MSL demo | | | | |
|---|---|---|---|---|
| N | Smpl dec (V/time) | | Simple (V/time) | |
| 100 | 14.21 | 0.03 | 11.72 | 0.05 |
| 500 | 13.618 | 0.16 | 11.744 | 0.16 |
| 2500 | 13.8244 | 0.78 | 11.7648 | 0.78 |
| 12500 | 13.8008 | 4.08 | 11.7658 | 4.1 |
| 62500 | 13.7835 | 20.79 | 11.7611 | 20.95 |
| 312500 | 13.7717 | 120.3 | 11.7609 | 133.14 |
| 500000 | 13.777 | 223.89 | 11.7616 | 250.21 |
| K9 demo | | | | |
| N | Smpl dec (V/time) | | Simple (V/time) | |
| 100 | 269 | 0.22 | 250 | 0.38 |
| 500 | 266.7 | 1.25 | 250 | 1.85 |
| 2500 | 266.24 | 6.15 | 250 | 9.47 |
| 12500 | 266.372 | 30.11 | 250 | 46.66 |
| 62500 | 266.282 | 154.89 | 250 | 234.45 |

TABLE I

| MSL demo | | | | | | |
|---|---|---|---|---|---|---|
| N | Smpl dec (V/fail/err (%)) | | | Pwc dec (V/fail/err (%)) | | |
| 100 | 9.59 | 0.19 | 32.5 | 10.9 | 0.25 | 23.3 |
| 500 | 9.738 | 0.054 | 28.5 | 9.732 | 0.224 | 28.53 |
| 2500 | 10.16 | 0.1716 | 26.5 | 11.2992 | 0.196 | 18.26 |
| 12500 | 10.4896 | 0.3692 | 24 | 11.9542 | 0.24488 | 13.27 |
| 62500 | 10.518 | 0.428096 | 23.7 | 12.156 | 0.249968 | 11.8 |
| 312500 | 10.5661 | 0.516698 | 23.27 | 12.1214 | 0.245722 | 12 |
| 500000 | 10.5825 | 0.58638 | 23.18 | 12.1814 | 0.240218 | 11.58 |
| K9 demo | | | | | | |
| N | Smpl dec (V/fail/err (%)) | | | Pwc dec (V/fail/err (%)) | | |
| 100 | 113 | 0 | 58 | 229.5 | 0.22 | 14.6 |
| 500 | 140 | 0 | 47.5 | 245.8 | 0.108 | 7.8 |
| 2500 | 115 | 0 | 56.8 | 250.8 | 0.0776 | 5.8 |
| 12500 | 111.708 | 0.00016 | 58 | 246.536 | 0.10792 | 7.4 |
| 62500 | 104.86 | 0 | 60.6 | 248.231 | 0.0916 | 6.8 |

TABLE II

navigating, tracking rocks and performing analysis of their structure. The health monitoring system assesses wheels behavior, and in particular a faulty mode that is due to a rock stuck in a wheel. Using the belief update at $10Hz$, the executive continuously assesses the possibility of triggering a floating branch referred to as 'rover wheel dance', that drives a little in reverse then forward to shake the rock off the wheel. Besides, the executive re-evaluates the plan at branch point when either the action cost model or the reward model change.

## VI. Conclusion

We have presented a simple strategy for the robust execution of contingent plans under uncertainty. It re-evaluates branch value functions at branch point, re-estimates branch conditions whenever necessary, and allows runtime insertion/replacement of plan portion with floating branches. This is the first step towards the development of powerful techniques for planning and execution under uncertainty. The MC approach is flexible and provides good results in any situation given that a sufficiently high number of samples is used. The algorithms presented are a baseline capability, and will be used later to assess the quality of more complex and focused approaches.

Other works include [15] that studies plan execution where there is uncertainty on the resource consumption. However, the executed plans are no contingent plans in the sense branch execution is not conditionned on decision functions over the resource state.

Future work includes pre-computing more advanced branch value functions at planning time [9], and researching methods for re-evaluating plans with concurrent actions.

## References


[1] L. Pedersen, M. Bualat, D. Lees, D. Smith, and R. Washington, "Integrated demonstration of instrument placement, robust execution and contingent planning," in *Proceedings of the 7th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[2] J. Bresina, R. Dearden, S. Ramkrishnan, D. Smith, and R. Washington, "Planning under continuous time and resource uncertainty: A challenge for ai," in *Proceeddings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

[3] R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington, "Incremental contingency planning," in *ICAPS-03: Proceedings of the Workshop on Planning under Uncertainty and Incomplete Information*, 2003, pp. 415–428.

[4] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[5] J. Boyan and M. Littman, "Exact solutions to time-dependent mdps," in *Advances in Neural Information Processing Systems 13*, 2000, pp. 1–7.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, B. Books, Ed. MIT Press, Cambridge, MA, 1998, 1998.

[7] S. Thrun, "Monte carlo POMDPs," in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen, and K.-R. Müller, Eds. MIT Press, 2000, pp. 1064–1070.

[8] N. Meuleau, R. Dearden, and R. Washington, "Scaling up decision theoretic planning to planetary rover problems," in *AAAI-04: Proceedings of the Workshop on Learning and Planning in Markov Processes - Advances and Challenges*, 2004.

[9] Z. Feng, N. Meuleau, and R. Washington, "Dynamic programming for structured continuous markov decision problems," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 2004.

[10] R. Washington and D. Lees, "Utility-based plan insertion for continuous resources," in *Proceedins of the IEEE 2004 International Conference on Robotics and Automation*, 2004.

[11] S. D., "On optimal and data-based histograms," *Biometrika*, vol. 66, pp. 605–610, 1976.

[12] I. A.J., "Recent developments in non parametric density estimation," *Journal of the American Statistical Association*, vol. 413, no. 86, pp. 205–224, 1991.

[13] F. Hutter and R. Dearden, "The gaussian particle filter for diagnosis of non-linear systems," in *Proceedings of the Fourteenth International Workshop on the Principles of Diagnosis*, Washington, DC, 2003.

[14] T. K. Willeke and R. Dearden, "Building hybrid rover models: Lessons learned," in *Proceedings of the Fifteenth International Workshop on the Principles of Diagnosis*, 2004.

[15] J. Gough, M. Fox, and D. Long, "Plan execution under resource consumption uncertainty," in *Proceedings of the Workshop on Connecting Planning Theory with Practice at ICAPS-04*, 2004, pp. 24–29.