DREAM: An Algorithm for Mitigating the Overhead of Robust Rescheduling

Jordan R. Abrahams, David A. Chu, Grace Diehl, Marina Knittel, Judy Lin, William Lloyd, James C. Boerkoel Jr.
Harvey Mudd College
Claremont, California 91711
{jabrahams, dchulasso, gdiehl, mknittel, julin, wlloyd, boerkoel}@hmc.edu

Jeremy Frank
NASA Ames Research Center
Mountain View, California 94035
jeremy.d.frank@nasa.gov

Abstract

Generating and executing temporal plans is difficult in uncertain environments. The current state-of-the-art algorithm for probabilistic temporal networks maintains a high success rate by rescheduling frequently as uncertain events are resolved, but this approach involves substantial resource overhead due to computing and communicating new schedules between agents. Aggressive rescheduling could thus reduce overall mission duration or success in situations where agents have limited energy or computing power, and may not be feasible when communication is limited. In this paper, we propose new approaches for heuristically deciding when rescheduling is most efficacious. We propose two compatible approaches, Allowable Risk and Sufficient Change, that can be employed in combination to compromise between the computation rate, the communication rate, and success rate for new schedules. We show empirically that both approaches allow us to gracefully trade success rate for lower computation and/or communication as compared to the state-of-the-art dynamic scheduling algorithm.

Introduction

Computing and executing schedules in uncertain environments is an enabling technology for many applications, such as cooperative teams of airborne, surface operating, or underwater robots. For instance, unmanned aerial vehicles (UAVs) have been used for missions ranging from collecting data on wildlife to monitoring wildfires. More complex missions have been proposed and, on small scales, demonstrated using UAV teams that communicate with each other over radio cross-link (e.g. Cesare et al.; Li et al. (2015; 2016)). Despite numerous advances in energy storage, battery life, and communication limitations can limit UAV mission duration and success (Quach et al. 2013). Conducting UAV missions, either independently or in teams, in the presence of uncertainty, may require rescheduling. Computing and communicating these new schedules consumes energy and time, which could shorten the mission's allowable length.

This paper focuses on providing effective, resource-aware control for applications involving uncertain environments that may challenge the success of a mission. However, a predefined schedule only uses the information that was available when it was created. As the mission progresses, uncertain events (e.g. unexpectedly long task durations) may disrupt the schedule. To exploit this new information, an algorithm must reschedule in response to these uncertain events.

Dynamic execution algorithms reschedule in response to new information to increase the chance that the mission succeeds. However, the current state of the art reschedules every time an uncertain event is resolved, which means that agents may frivolously compute and communicate a new schedule despite minor performance gains. Frequent computation and communication is problematic when agents may have limited energy (e.g. batteries) or communication bandwidth, and wise use of these resources is critical for mission success. Unnecessary computation can waste energy, thereby reducing the chance of mission success. Further, agent communication can also be intermittent, slow, or unreliable; reducing recomputation in such circumstances can increase the risk that an agents' constraints can't be satisfied .

We address these challenges with two new strategies for mitigating rescheduling overhead:

- *Allowable Risk* reschedules more often when schedules are more risky, and less often when schedules are less risky; and,
- *Sufficient Change* sends a new schedule only when it significantly changes the predicted probability of success.

These strategies complement each other, and can be combined into into a new algorithm we call the *Dynamic Robust Execution Algorithm with Mitigation (DREAM)*. We conduct an empirical evaluation of the trade-offs between rescheduling frequency and success rate, showing that Sufficient Change and Allowable Risk can gracefully trade reductions in success rate for significant reductions in how often we need to compute and communicate new schedules. These trade-offs are systematically demonstrated on two existing datasets with different characteristics.

Background

As a motivating example, consider a wildfire surveillance coordination problem involving two UAVs, A and B. The UAVs have a preliminary plan, and are controlled by a remote base-station that can update the plan as the mission proceeds. The agents must exchange infrared images that they have already taken of the wildfire, but due to limited communication range, they must first travel from their current respective locations to a pre-negotiated rendezvous point. UAV A's and UAV B's navigation tasks take are normally distributed, and take on average 6 and 2 seconds, respectively; the uncertainty is due to adverse flying conditions and localization error. UAV B, which starts out closer to the rendezvous point, must also take an image at the new location before establishing a communication link. Acquiring the image takes 2 seconds. Finally, to establish a successful communication link, both UAVs have to be within range of the same point within 2 seconds of each other. The makespan constraint for this entire task is 10 seconds. As shown in Figure 1, we can model and solve this problem using Probabilistic Simple Temporal Networks (PSTNs), which we define next.

Temporal Networks

A *Simple Temporal Network* (STN), $S = (T, C)$, consists of a set $T = \{t_0, t_1, \ldots, t_n\}$, where each timepoint t_i represents the time at which a distinct event happens, and a set C of binary constraints on events in T. These constraints are of the form $t_j - t_i \leq b_{ij}$, for some $b_{ij} \in \mathbb{R}$ (Dechter, Meiri, and Pearl 1991). The two constraints between t_i and t_j can be written concisely as $t_j - t_i \in [-b_{ji}, b_{ij}]$. STNs are often encoded as directed graphs, where events are vertices and constraints are edges. A *schedule* is an assignment of time values to events such that all constraints are satisfied. An STN is considered *consistent* if it has at least one schedule. One drawback of STNs is that they cannot formally represent durational uncertainty between events. Since the physical world is inherently uncertain, accounting for uncertainty in our representation allows it to better model reality.

Simple Temporal Network with Uncertainty (STNU) In a *Simple Temporal Network with Uncertainty* (STNU) (Vidal and Ghallab 1996), the set of constraints C is divided into two disjoint subsets, called C_R, the set of *requirement* constraints, and C_C, the set of *contingent* constraints. Requirement constraints are identical to standard STN constraints. A contingent constraint represents that the time that elapses from t_i to t_j, given by $\beta_{ij} \in [-b_{ji}, b_{ij}]$, is chosen by an uncontrollable process and is unknown prior to execution.

An event whose incoming edges are all requirement edges is known as an *executable* timepoint, because the agent executing the schedule controls when it happens, or *executes*. A timepoint with an incoming contingent edge is known as a *contingent* timepoint, since it happens automatically some time after the timepoint that initiates the contingent constraint. When a contingent timepoint happens it is said to be *received*. We call the set of contingent timepoints T_C, and the set of executable timepoints T_X.

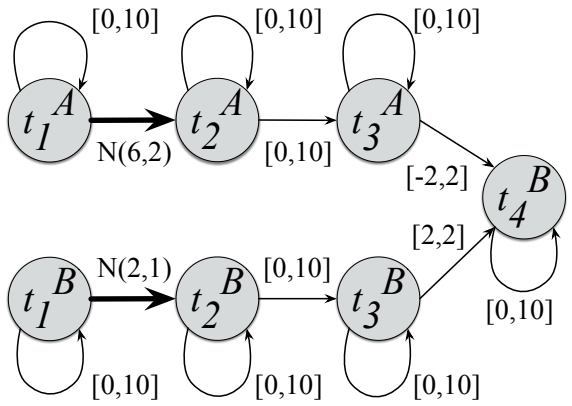


Figure 1: Our example problem illustrated as a PSTN.

Controllability Algorithms for STNUs STNUs are *strongly controllable* if times can be selected for each executable timepoint such that, for *all* possible contingent timepoint outcomes, *all* requirement constraints are satisfied. Not all STNUs satisfy this restrictive property. STNUs are *dynamically controllable* if we can find a scheduling strategy, where some decisions are contingent on the outcomes of uncertain events during execution, that guarantees success. Dynamic controllability is a less restrictive property than strong controllability; the decision to execute a timepoint can be deferred until information about uncontrollable timepoints is known (Vidal and Ghallab 1996; Vidal 2000; Morris, Muscettola, and Vidal 2001; Morris and Muscettola 2005; Morris 2014; Akmal et al. 2019).

Probabilistic Simple Temporal Network (PSTN) A *Probabilistic Simple Temporal Network* (PSTN) extends an STNU by adding information about the uncertain processes that govern contingent edges. For a PSTN's contingent edges, the time that elapses from t_i to t_j is chosen by a random variable X_{ij}, whose value is determined at execution by some Probability Density Function (PDF) P_{ij} (Tsamardinos 2002; Fang, Yu, and Williams 2014; Brooks et al. 2015). Since contingent edges in PSTNs are governed by unbounded probability distributions, they cannot be strongly controllable. However, as we will later discuss, some algorithms still use the idea of strong controllability to solve PSTNs.

Example Problem The PSTN representation of our running UAV example problem is shown graphically in Figure 1. Each vertex represents a timepoint. The agent (A or B) to which a particular timepoint corresponds is indicated by the superscript. Timepoints denote the start and end times of an agent's task (e.g. deciding when to start or stop moving). Directed edges represent temporal constraints and are labeled with the range of time that is allowed to elapse between the occurrence of the events represented by the source and target timepoints. Thick edges represent contingent constraints, and straight, slim edges represent requirement con-

straints. The notation $N(\mu, \sigma)$ in our example problem indicates a normal probability density function with mean μ and standard deviation σ. A's navigation task is represented by the edge between timepoints t_1^A and t_2^A, (and similarly for B). B's 2 second image acquisition task is represented by the edge between t_3^B and t_4^B. The event where a communication link between A and B is established is represented by the edge t_3^A and t_4^B. All tasks must be completed within 10 seconds, shown by the 'self' constraint [0,10] for each timepoint.

Controllability Algorithms for PSTNs

Previous work in PSTNs (Tsamardinos 2002; Fang, Yu, and Williams 2014; Santana et al. 2016) has focused primarily on algorithms for generating a control policy up-front. Most prior work in this area focuses on a variant of strong controllability, that is, finding a schedule consisting of assignments of controllable timepoints that satisfies requirement constraints subject to some risk, or alternately, maximizes robustness. Strong controllability is a more difficult to satisfy property than dynamic controllability; a strong schedule requires choosing a single time to execute every controllable timepoint that works no matter what the outcome of the uncontrollable timepoints, while a dynamic strategy can react to observations of the uncontrollable timepoints.

By contrast, this paper investigates algorithms that reschedule actively during execution. While dynamic controllable policies describe how to execute controllable timepoints in response to observed uncontrollables, rescheduling can revise the entire schedule continuously. Rescheduling algorithms are an improvement over strong controllability, as noted previously. However, expensive rescheduling and communication of schedules may inefficiently use resources compared to the payoff in robustness.

Early Execution Early Execution is a naïve algorithm for deciding when to execute the timepoints in a PSTN. As its name implies, it executes timepoints as soon as they can be executed–when they are both *live*, meaning that they are within their acceptable time range, and *enabled*, meaning that all predecessor timepoints have been executed.

The Static Robust Execution Algorithm While algorithms like Early Execution can be effective in practice, they are agnostic as to the impact of uncertainty on performance. In our UAV example problem, if both agents start navigating as soon as possible, it is highly likely that UAV B will arrive at its destination more than 2 seconds before UAV A, resulting in failure. To maximize the probability of success, UAV B should wait before navigating. The Static Robust Execution Algorithm (SREA) is motivated by this limitation (Lund et al. 2017). SREA is an *offline* approach that tries to address this limitation by maximizing *robustness*, the probability that for a given strategy, all events are executed without violating constraints (Brooks et al. 2015). Robustness is the complement of risk (Fang, Yu, and Williams 2014).

In order to maximize robustness, SREA transforms the input PSTN into a strongly controllable STNU while approximately minimizing the probability of failure. SREA sets a maximum probability α that each contingent timepoint in the original PSTN violates its contingent edge's bounds because it occurs too early or too late. (α is *identical* for all contingent constraints.) This makes $1 - \alpha$ the minimum probability mass of each contingent edge captured by a corresponding interval in the STNU. To find the optimally robust schedule, SREA does a binary search over α. For each α, it uses a linear program (LP) to maximize the probability mass captured by the interval over each contingent timepoint. In a sense, SREA maximizes the probability that uncertain events will occur during these intervals. From this, SREA produces a schedule that guides an agent on when it should execute the events to give uncertain events the best chance of falling within these precomputed intervals. In our running example, SREA would constrain UAV B to wait before navigating to maximize the probability that the arrival times of both agents overlap. SREA is similar to the approach of Santana et al. (2016), which uses a piecewise constant approximation of the PDF, allowing the use of a different LP formulation than that of SREA. A different static approach due to Fang, Yu, and Williams (2014) uses a nonlinear solver to allocate risk in a more globally optimal way rather than rely on the heuristics employed by LP approximations.

SREA is good at using information about uncertainty to maximize the probability of success. However, it can fail when uncertain timepoints fall outside of their designated intervals during execution. In addition, like Strong Controllability, on which it is based, SREA is limited because it cannot re-optimize risk on constraints or reschedule the execution of controllable timepoints when new real-time information, such as the actual time of an uncertain event, is gained. This is because SREA is a *static* algorithm, in that it does not change the schedule in real-time.

The Dynamic Robust Execution Algorithm Dynamically updating the guiding schedule can be beneficial in situations with uncertain events. The Dynamic Robust Execution Algorithm (DREA) is an *online* algorithm that builds on SREA with the goal of maximizing robustness by adding the ability to incorporate new information during execution (Lund et al. 2017). It creates an initial schedule by running SREA and uses it to guide execution. As contingent timepoints are received, DREA updates the PSTN with this new information, and calls SREA to create a new schedule.

Related Approaches

In general, this class of problem can also be posed and solved as a (Partially Observable) Markov Decision Problem ((PO)MDP). Large, hard to solve MDPs have been addressed by 'on-line' methods that partially solve the MDP at each step (Dean et al. 1995; Kearns, Mansour, and Ng 2002; Péret and Garcia 2004). These methods address the problem of trading computation time for solution quality. The method of Wu, Zilberstein, and Chen (2011) considers how to reschedule and communicate new schedules in the presence of restricted communication. For our work, we limit the expressiveness of the problem under consideration to PSTNs. While tractable, the PSTN framework still exposes

the fundamental problem of deciding when to reschedule and communicate changes of schedules between agents.

Mitigating DREA's Overhead

DREA has a significantly higher success rate than both Early Execution and SREA on the set of randomly generated benchmark PSTNs in Lund et al. (2017). However, this high success rate comes with the cost of large amounts of rescheduling. In many scenarios, including our UAV example, computing and sending new schedules is costly, so DREA may have an undesirably high overhead.

With this in mind, we have investigated two possible methods for limiting when DREA reschedules and communicates, without drastically diminishing success rate. The first method, *Allowable Risk*, makes rescheduling (and thus both computation and communication) frequency proportional to the level of uncertainty that has been resolved in the problem. *Sufficient Change* seeks to reduce communication frequency by only communicating when the new schedule has a significant change in the amount of risk. The two approaches complement each-other. We have combined them into a new version of DREA with control parameters that allow us to vary the trade-offs.

Algorithm 1: DREAM

Input : A PSTN, S
Input : A min success threshold, $0 \leq m_{AR} \leq 1$
Input : A min change threshold, $0 \leq m_{SC} \leq 1$
Var : Current time, now
Var : Next Timepoint Encountered, t
Var : A guide STN, G
Var : Risk of guide, α
Var : Contingent Edge Counter, k

$(\alpha, G) \leftarrow \texttt{SREA}(S)$;
$k \leftarrow 0$;
while $\texttt{Consistent}(S)$ **and not** $\texttt{AllExecuted}(S)$ **do**
  `// Next dispatched/received event`
  $(now, t) \leftarrow \texttt{AdvanceToNextEvent}(S)$;
  if $t \in T_C$ **then**
    $S.update(t = now)$;
    $G.update(t = now)$;
    $k \leftarrow k + 1$;
    `// Check Allowable Risk`
    if $(1 - \alpha)^k \leq m_{AR}$ **then**
      $(\alpha', G') \leftarrow \texttt{SREA}(S)$;
      `// Check Sufficient Change`
      if $|\alpha' - \alpha| \geq m_{SC}$ **then**
        $\alpha \leftarrow \alpha'$;
        $G \leftarrow G'$;
        $k \leftarrow 0$;
  else
    $S.update(t = now)$ $G.update(t = now)$;

We present these algorithms combined as *Dynamic Robust Execution Algorithm with Mitigation* (DREAM). It acts similar to DREA, except when a contingent timepoint is received, DREAM more judiciously decides first whether or not to reschedule, and then whether to communicate. Allowable Risk and Sufficient Change use thresholds m_{AR} and m_{SC} respectively to make these decisions. The details of the DREAM algorithm are shown in full in Algorithm 1.

Reducing Rescheduling: Allowable Risk

As risk accumulates during execution, the current execution guide STN has a lower chance of succeeding, and so rescheduling becomes more valuable. Allowable Risk (AR) decides how many uncertain events can occur before rescheduling while maintaining an acceptably high overall probability of success. Then, it dispatches events according to the guide, updating the guide by rescheduling after the previously computed number of uncertain events have occurred. This strategy reduces the number of reschedules while still guaranteeing the risk will not exceed a threshold without rescheduling. Thus, AR reschedules less often in cases of low risk, sparing excess computation.

AR is detailed in the first part of the DREAM algorithm. It requires an input $0 \leq m_{AR} \leq 1$ to represent the minimum robustness threshold. As a proxy for probability of failure, α ties rescheduling to the riskiness of the guides generated by SREA. AR first finds the smallest integer value of n such that

$$(1 - \alpha)^{n+1} > m_{AR}$$

which is equivalent to

$$n \geq \log_{(1-\alpha)}(m_{AR}) - 1.$$

AR dispatches events until the received (uncontrollable) event counter k exceeds n, then reschedules. When the algorithm reschedules, AR resets k to 0. Rescheduling will also produce a new α and guide STN G, leading to a new value for n (if the schedule has a sufficient change in α, discussed below). Thus, if SREA generates a schedule with high α, AR will reschedule sooner than if SREA had generated a schedule with low α.

Since m_{AR} is the minimum probability of success allowed without rescheduling, a higher threshold could lead to frequent rescheduling, resembling DREA. However, a lower threshold could lead to low success rate from too little rescheduling, performing like SREA.

Reducing Communication: Sufficient Change

Sufficient Change (SC) determines whether the algorithm will actually communicate the new guiding schedule to the agents. The concept underlying Sufficient Change is to only communicate a new guiding schedule G' if it has a significant change in the chances of success compared to the current guide G. This SC determination prevents sending out new guides that provide little change in robustness to the agents. Note, SC does not reduce the amount of rescheduling, since it requires the SREA algorithm to be run prior to deciding whether or not to communicate the new schedule.

When DREAM runs SREA to generate a new guide G, it assigns an interval to each uncertain edge that captures

some of the probability mass of that edge, and the remaining probability mass uncaptured is α. SREA optimizes for the smallest α across all contingent edges while still maintaining consistency. Thus α acts as a proxy for the risk level of the guide overall.

In Algorithm 1, G' represents a new updated guide produced by DREAM. Since G' is more recent than the current guide G, G' will take into account the assignment of past contingent timepoints. Because of this, G' will generally have lower risk than G since there is less uncertainty remaining in the network. However, in cases where a contingent timepoint is assigned early or late unexpectedly and thus falls out of the SREA capture range, then there are cases where α' may be higher than α. It is crucial to reschedule in these cases, as they may require large changes to the execution times to improve robustness.

If the absolute difference between α' and α exceeds the SC threshold m_{SC}, then the new guide STN G' will replace the current guide STN G. α is also replaced with α'.

SC's threshold influences how often the algorithm reschedules. If the threshold is low, we expect the algorithm will reschedule often, resembling DREA. Alternatively, if the threshold is high, the algorithm will reschedule less often, performing like SREA. In between extreme values, we expect to see a trade-off, where we reduce communication but also reduce robustness as the threshold approaches 1.

DREAM Execution on Example Problem

To illustrate how the DREAM algorithm executes and the influence of parameter settings, we step through a hypothetical execution using the example problem introduced in Figure 1. First, SREA is applied to the original PSTN, resulting in a guide STN and corresponding α. Next, the guide STN is used to dispatch scheduling decisions. This would result in UAV A and UAV B departing at the earliest times suggested by the guide STN. Since both of these departures are executable timepoints (not contingent) they have no impact on DREAM's decision to reschedule. Suppose UAV A arrived earlier than expected, e.g. after 2 seconds instead of the expected 6 seconds. DREAM checks if $(1-\alpha)^1 < m_{AR}$. For the sake of argument, suppose this condition is true (the old schedule is now too risky given the early arrival time). DREAM reschedules, and a new guide STN G' and per-link risk α' are calculated; UAV B's image acquisition and the communication events are scheduled earlier to account for A's early arrival. The new schedule is communicated to the UAVs if $|\alpha' - \alpha| > m_{SC}$, which would be likely for reasonably low (and therefore sensitive) m_{SC} values.

Empirical Evaluation

The ultimate goal of our analysis is to understand and explore the trade-off between reschedules/communications and success rate in our algorithms. To do this, we constructed an event-based Monte-Carlo simulator to empirically test the performance of DREAM[1]. The simulator uses pre-generated PSTNs and draws contingent edge durations from normally distributed PDFs. To mimic real-world uncertainty, the durations of contingent constraints are only revealed to the scheduler once the respective contingent timepoint is received. Guide schedules tell an agent when it should execute all remaining (unexecuted) timepoints. We track the success rate, the total solve time per simulation, the reschedule rate, and the communication rate on each STN across all trials.

[1]All simulator code and problem instances are available for download from https://github.com/HEATlab/DREAM

DREA Characteristics	
Robustness	$33.64\% \pm 1.5\%$
Runtime	$3.68\text{s} \pm 0.49\text{s}$ per sim
Reschedule Rate	4.63 ± 0.09 per sim
Communication Rate	3.44 ± 0.12 per sim

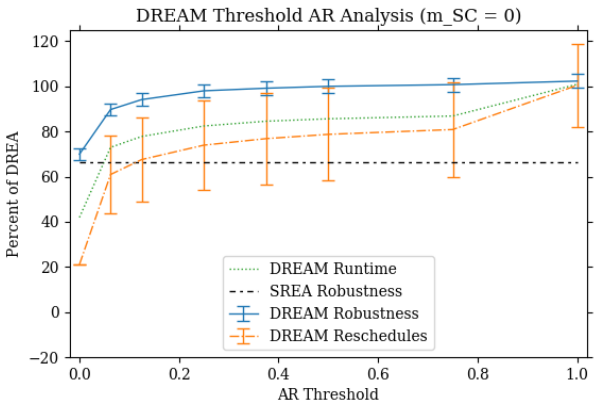
Table 1: Characteristics of DREA on the problem set used in this paper. Errors in this table are 1 standard error.

We tested DREAM on 540 PSTNs taken from Lund et al. (2017)[1], each containing 21 events and at least 2 agents. 100 simulations were run on each PSTN with a different random seed. A simulation returns execution success or failure. PSTNs which failed across all simulations for every algorithm were omitted in further analysis. Of the 540 test problems, 436 were solved by at least one of the algorithms we tested; all of the statistics below are averages over these 436 problems.

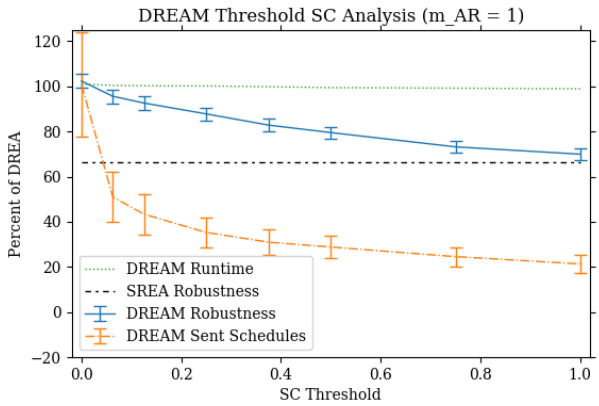
We tested a parameter sweep across combinations of m_{AR} and m_{SC}, both ranging from 0 to 1 with logarithmic density (e.g. $m_{AR} = \{1, 0.5, \ldots, 0.0625, 0\}$). By varying the thresholds, we gain more insight into the trade-offs between how frequently schedules are recomputed/communicated and schedule success rate. We compare the success rates and number of reschedules of our algorithms to those of DREA and SREA, which both represent the state-of-the-art dynamic and static robust scheduling approaches, and also the extreme versions of DREAM that always/never reschedule respectively. This analysis can help determine which algorithm and threshold are most appropriate for applications with various computational needs.

First, we compare the performance of AR and SC in isolation, by fixing $m_{SC} = 0$ while varying m_{AR}, and fixing $m_{AR} = 1$ while varying m_{SC}. In Figure 2, we show how varying AR and SC affects robustness, runtime, and communication frequency. Both of the plots show how DREAM performs relative to the state of the art execution strategy, DREA. Each data point in Figure 2 has a surrounding 95% confidence interval, though runtimes lack these intervals for plot clarity.

Since all data points in the following two plots are proportional to DREA's performance, Table 1 provides DREA's performance for reference. An important point to note is that DREAM's communication rate is consistently lower than DREA's from (Lund et al. 2017) because the SREA algorithm will occasionally fail to find an acceptable α which solves the internal LP, but the agent will continue to attempt to reschedule until it reaches an invalid timepoint assignment. This characteristic persists in DREAM as well.



(a) Allowable Risk (AR) threshold variation results



(b) Sufficient Change (SC) threshold variation results

Figure 2: Simulated results for Allowable Risk (AR) and Sufficient Change (SC) on instances from (Lund et al. 2017). We plot the success rate, the mean runtimes, reschedules (a proxy for both computation and communication for AR) and communications (for SC) relative to DREA. We also plot SREA's robustness to show improvement over the baseline.

Impact of Allowable Risk

From Figure 2a, we make three significant observations. The first is that at $m_{AR} = 1$, DREAM achieves the highest robustness (success rate), but also the highest reschedule rate and runtime. Additionally, when $m_{AR} = 1$ and $m_{SC} = 0$, we do reach nearly identical performance characteristics as DREA. We also note that as m_{AR} approaches 0, we see DREAM performs similarly to raw SREA. This is expected, as an AR threshold of 0 would indicate that we never reschedule beyond the start of the problem. Thus, both of the endpoints match our expectations. The third observation is that there are several values of $m_{AR} > 0.25$ where DREAM has a large improvement in runtime and reschedule rate with a minor robustness penalty. The improvement here shows that DREA reschedules excessively, and that some reschedules in DREA had negligible impact.

Impact of Sufficient Change

Figure 2b shows the same analysis, but instead now focusing on SC threshold and communication rate. We fix $m_{AR} = 1$ (the AR threshold with the highest reschedule rate); for these runs, we reschedule at every received contingent time point, as DREA normally would. The runtime in Figure 2b matches DREA regardless of the value of m_{SC}, because we save nearly no computation. However, we see a steep decrease in communications at $m_{SC} = 0.0625$, with only a marginal drop in robustness. Similar to the pattern observed for AR, Figure 2b's endpoints line up with the DREA and raw SREA execution strategies. However, the curve direction is reversed, as a lower SC threshold is more strict, i.e. requiring more rescheduling, while a higher AR threshold is more strict. With a low m_{SC}, what we classify as a "sufficient" change in α is more broad, leading to more sending of guide schedules.

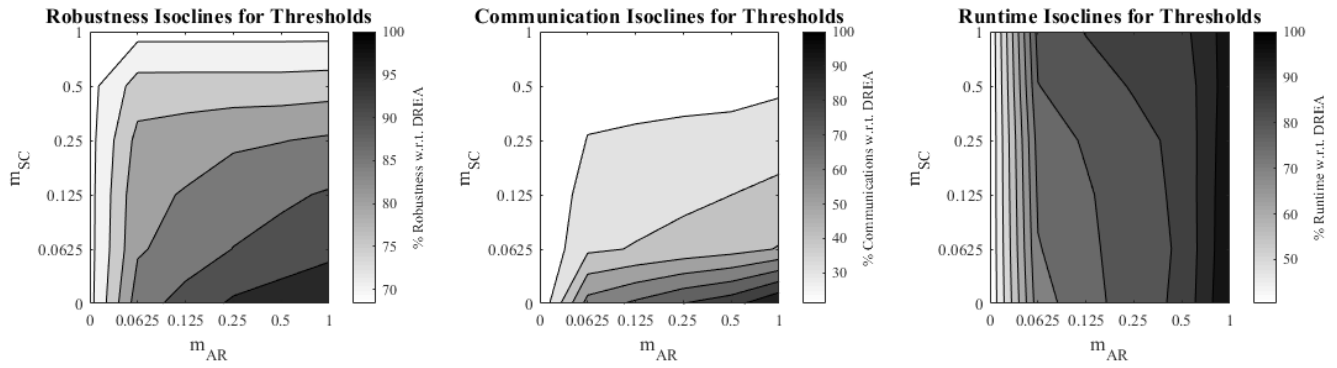
One unexpected result shown in this plot is the shape of the robustness curve. Compared to the AR, we see a much more gradual trade-off between m_{SC} and robustness; changing m_{SC} within the full range $[0, 1]$ leads to significant changes in success rate. This does not hold for the number of communications, as $m_{SC} > 0.5$ leads to only minor improvements here.

Exploring the Landscape of Trade-offs

While we have analyzed how both thresholds of DREAM affect results in isolation, we are ignoring an important feature of this algorithm. DREAM is able to use both the AR and SC filters simultaneously. In order to properly evaluate DREAM's performance, we must also analyze how combinations of thresholds affect our metrics. To analyze these trade-offs, we conducted a parameter sweep over a logarithmic-like grid for (m_{AR}, m_{SC}) values. Each threshold value was chosen from the set $\{0, 0.0625, 0.125, 0.25, 0.5, 1\}$. From Figures 2a and 2b, we expect DREAM to react similar to DREA when $(m_{AR}, m_{SC}) = (1, 0)$.

We show the results of this parameter sweep in Figure 3. These plots are filled contour plots which focus on robustness, communication, and runtime for each threshold combination. As with Figure 2, each dependent variable is plotted proportional to DREA (100% indicates performing identically to DREA).

From these plots, we notice several important features of this threshold landscape. From Figure 3a, we find that the only threshold pair which has the same robustness performance as DREA is when $(m_{AR}, m_{SC}) = (1, 0)$, as expected. However, we find that when $m_{AR} > 0.25$ and $m_{SC} < 0.0625$, DREAM can trade much substantially lower schedule communications and computation runtimes for only slightly lower robustness levels. For example, the combination of $(m_{AR}, m_{SC}) = (0.5, 0.0625)$ has a relative robustness of 98%, a relative communication frequency of 47%, and a relative runtime of 81% compared to DREA on the same data set. This demonstrates that DREAM can trade a small loss in robustness for large performance gains



(a) Robustness contour plot. (b) Communication contour plot. (c) Runtime contour plot.

Figure 3: Contour plots showing how threshold values affect robustness, communication, and runtimes of DREAM on instances from (Lund et al. 2017). Note we use a logarithmic scale to show more detail near smaller values. Also note that contour lines are interpolated, and hence may not align perfectly with the raw data for each parameter combination.

in other metrics.

Another important insight we can gather from this visualization is how m_{AR} and m_{SC} combine to reduce the need to communicate new schedules. m_{SC} has diminishing returns as it increases beyond 0.125, as robustness declines steadily while maintaining similar communication frequency. On the other hand, communication is much more sensitive to changes in m_{AR}, and small changes in Allowable Risk threshold below 0.0625 can lead to a rapid communication reduction.

Finally, Figure 3c shows that m_{SC} has minor (if any) effect on runtime, while runtime is quite sensitive to changes in m_{AR} for (especially for low values). These results match our expectations, and reinforce the data shown in Figure 2.

Finding Optimal Threshold Values

From the data in Figure 2, we can find the threshold values that provide the highest robustness while also minimizing the amount of either reschedules or communications.

When we fix m_{SC} to its most sensitive value (that is, $m_{SC} = 0$), we can identify an "optimal" value for m_{AR} by comparing robustness and reschedule rate over all m_{AR} values. To achieve the most robustness for the least rescheduling, we define the optimal m_{AR} to be where we maximize the ratio between robustness improvement from a static strategy (SREA) and the number of *reschedules*, since AR controls rescheduling rate. If we denote the robustness of SREA by R_S, the robustness of DREAM with a specific value of m_{AR} by $R_D(m_{AR})$, and the number of reschedules achieved by DREAM for a specific value of m_{AR} by $S_D(m_{AR})$, then we seek

$$\max_{m_{AR}} \frac{R_D(m_{AR}) - R_S}{S_D(m_{AR})}.$$

Note, SREA never reschedules and so does not appear in the denominator of this ratio. We know R_S from Figure 2, and the remaining components of this optimization from our data. From the threshold values we tested, we found the best performance to be $m_{AR} = 0.250$, which only requires on average 74% of the reschedules of DREA.

Similarly, we define the optimal m_{SC} as the value where we maximize the ratio between robustness improvement and the number of *communications* given $m_{AR} = 1$. We focus on communication rate here because varying m_{SC} does not change the amount of computation an agent does. In this case we denote the number of communications achieved by DREAM for a specific value of m_{SC} by $C_D(m_{SC})$, and we seek

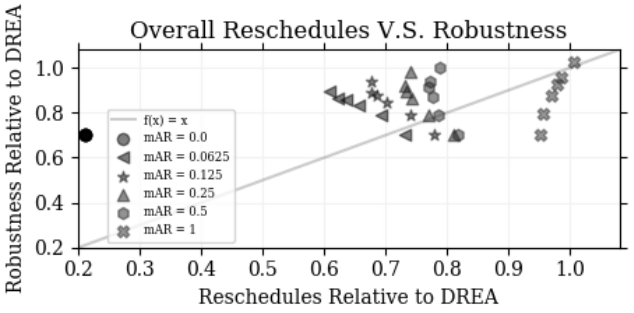
$$\max_{m_{SC}} \frac{R_D(m_{SC}) - R_S}{C_D(m_{SC})}.$$

From the threshold values we tested, we found the best performance to be $m_{SC} = 0.250$ when individually analyzed, which sends just over 35% of the schedules that DREA does.

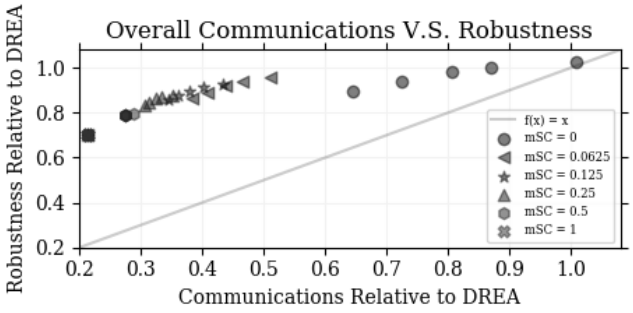
These combinations give robustness values of 98.1% of DREA and 87.8% of DREA, respectively. We do note that in the case of the AR threshold, any m_{AR} value between 0.0625 and 0.5 seems to give metric values which are similar enough to be within error. Therefore, there exists a plateau in trade-off quality when varying the AR threshold which is not present in SC threshold variation.

While we have found the optimal threshold settings in isolation, for a complete description of DREAM, we must attempt to find the combination of thresholds which provides the most improvement on robustness for the least overhead. To do this, we generated two scatter plots that show how robustness is sacrificed as we reduce either the number of reschedules (Figure 4a) or communications (Figure 4b). All value presented here are relative to DREA (that is, normalized relative to the original DREA performance). We prefer trade-offs that lead to points that appear in the upper left corner of the space. Thus, the points that appear on left/top side of the convex hull formed by these points represent various optimal trade-offs between reducing computational overhead without sacrificing robustness.

Observe that most points in Figure 4a and *all* points in Figure 4b appear above the curve $f(x) = x$, which would represent a 1-for-1 trade-off. This is extremely encouraging, since it means regardless of how we set m_{AR} and m_{SC}, the relative reduction in computational overhead outpace the rel-



(a) Reschedules vs. Robustness, stratified by m_{AR} values.



(b) Communications vs. Robustness, stratified by m_{SC} values.

Figure 4: Scatter plots showing robustness as a function of the reschedules (4a) and communications (4b) relative to DREA for every point in our parameter sweep of m_{AR} and m_{SC} on instances from (Lund et al. 2017). Here, we want points that appear in the upper (low robustness loss) left corner (high resource savings), so any point that appears above the curve $f(x) = x$ represents a net positive trade-off.

ative losses in robustness.

Further, we note that in Figure 4a, in some cases the points appear to form "stacks". This is because, for given setting m_{AR}, varying m_{SC} will have no impact on how often DREAM reschedules. To illustrate this, plot points in 4a for the same value of m_{AR} are plotted with the same plot symbol. However, since new schedules are communicated less often, varying m_{SC} will negatively impact robustness. This behavior is evident in Figure 3c, and is replicated in 4a. As expected, the points that optimally trade reduced reschedules for robustness are those with high m_{SC} values. Figure 4b, on the hand, demonstrates that overall, DREAM very gracefully trades reductions in communication for reductions in robustness. In fact, most points are closer to appearing on the optimal face of the convex hull than they are to the curve $f(x) = x$. We chose to use the same plot-point symbol for identical values of m_{SC} in Figure 4b to show that m_{SC} generally but not always predicts robustness.

Comparing the individual AR and SC threshold performances also reveals how riskiness of a schedule changes over time. Figure 4b shows that communications changes greatly between $m_{SC} = 0$ and $m_{SC} = 0.0625$. This stark difference indicates that roughly half of the schedules sent by DREAM have only minor α changes. A small change in α suggests that the new schedule produced by DREAM was similar to the previous schedule (or at least the captured probability was similar). However, we do notice a decrease in robustness by removing half of those communications. Therefore, even small updates are vital for maximizing the success of an execution problem.

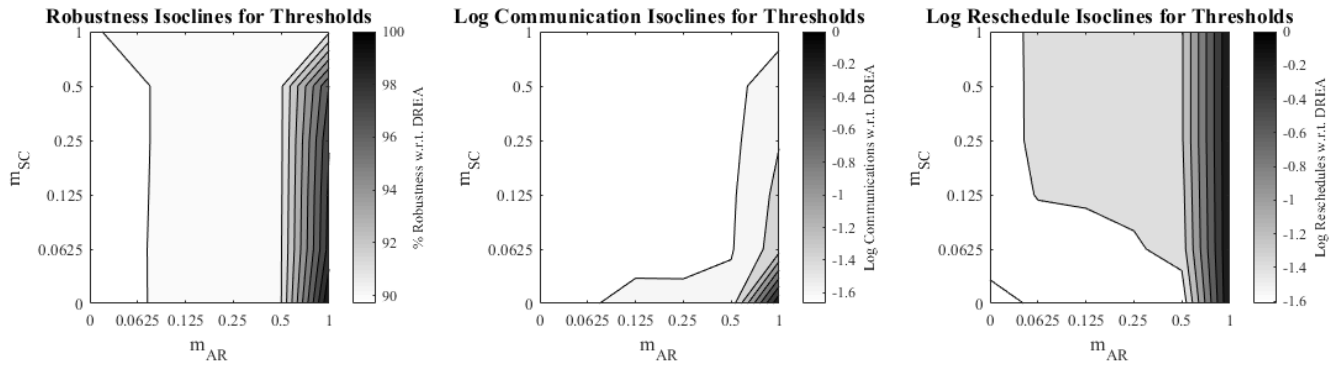
Evaluating DREAM on ROVERS Dataset

To show that DREAM provides similarly graceful trade-offs on other problems, we conducted the same empirical evaluation on a subset of the ROVERS dataset from Santana et al. (2016). The ROVERS problem set is a set of 4380 Probabilistic Simple Temporal Networks with Uncertainty (PSTNUs). PSTNUs augment the PSTN formulation by designating some contingent constraints as having only minimum and maximum time bounds, but no underlying probability distributions (as in STNUs). This is in contrast to PSTNs, where all contingent edges have some underlying probability distribution. Since DREAM only functions on PSTNs, we convert each PSTNU to a PSTN by modeling the simple temporal constraints with uncertainty as contingent edges with uniform distributions.

With these modifications, we conducted an AR/SC threshold grid search to match our results shown in Figure 3. Due the size of the ROVERS PSTNs and simulation solve time constraints, we were limited to the first 73 PSTNs (a small subset) of the ROVERS dataset. These PSTNs range between 71 to 223 events, and about one quarter of edges in each were contingent. We ran 50 simulations on each PSTN to approximate the robustness associated with our tested threshold settings, with same logarithmic grid pitch used Figure 3.

The results of this grid search are shown the three contour plots in Figure 5. We represent the number of communications and reschedules with logarithmic scale, as at $(AR, SC) = (1, 0)$, the number of schedule sends and the number of reschedules are about 40 times more than the lowest counts. These plots highlight several crucial differences between the Lund et al. problems and the Santana et al. ROVERS problems. The first major difference is that SREA, with no reschedules at all, performs surprisingly well in comparison to DREA on ROVERS. This is highlighted by observing barely any increase in robustness for $m_{AR} < 1$ in Figure 5a: we can achieve just over 90% of DREA's performance on these problems with no rescheduling at all. We observe that many of ROVERS problems had every simulation succeed, or every simulation fail. This hints that many of the ROVERS problems are strongly controllable, and many have no valid schedules or have very improbable schedules; this is consistent with the results reported by Santana et al. (2016) that 2840 of 4380 instances are strongly controllable, but in 911 of these cases, at least one probabilistic duration squeezed to a single value.

We also see a lack of tunability in AR threshold in Figure



(a) Robustness contour plot for ROVER data. (b) Comm. contour plot for ROVER data. (c) Reschedule contour plot for ROVER data.

Figure 5: Contour plots showing how threshold combinations change performance on the ROVER dataset from Santana et al. (2016). Again, our axes are not linearly scaled, and were sampled at the same density as Figure 3. We note however the stark contrast which occurs at $m_{AR} = 1$, which is the only location where we see comparable robustness values to DREA. Due to the orders of magnitude difference between $m_{AR} = 1$ and all other m_{AR} settings, we show $\log_{10}$-scale plots for number of communications and number of reschedules.

$5a$. Only at $m_{AR} = 1$ do we see a significant improvement [2] in robustness over SREA. This is a suprising result, and suggests two possible explanations. The first explanation is that binary search conducted by SREA is unable to find a valid schedule, which in turn means DREAM could not update the agent with an improved schedule, causing more failures, and a performance similar to SREA. The second explanation is that because AR measures the accumulation of risk as agents execute, the PSTNs lack notable accumulations of risk. If the α of the current schedule is very small DREAM will not reschedule even for high values of m_{AR}. However, as soon as $m_{AR} = 1$, DREAM will reschedule at every contingent timepoint regardless of a schedule's α. This explains why there are order-of-magnitude more reschedules when $m_{AR} = 1$, as shown in Figure $5c$.

Despite the tiny tunable range of m_{AR} threshold, we see strong trade-offs with some m_{SC} thresholds, as highlighted in Figure 5b. As long as $m_{AR} = 1$, and $0 < m_{SC} < 0.5$, we see 97% to 99% performance of DREA with 3% to 5% the number of schedule sends. This means that at least 95% of the schedules sent by DREA had almost no effect on success rate. This shows that DREAM provides a significant reduction on the amount of communication required with almost identical success rate and also underscores the value in having a tunable algorithm like DREAM that can help achieve maximum robustness while minimizing computational and communication overheard across problem domains with extremely different characteristics.

Conclusion and Future Work

In this paper, we augment DREA, a state-of-the-art dynamic schedule execution algorithm, to maintain its high success rate while simultaneously reducing its high amount of communication and computation. To this end, we propose two techniques to mitigate these costs: Allowable Risk (AR) and Sufficient Change (SC). Combined, these complementary augmentations to DREA form our new approach: Dynamic Robust Execution Algorithm with Mitigation (DREAM).

DREAM is a fully tunable hybrid between SREA and DREA. Our exploration of DREAM shows a clear trade-off between rescheduling and success. Robustness of schedules generally degrades with higher m_{SC}, which reduces DREAM's willingness to communicate novel schedules. For fixed m_{SC}, varying m_{AR}, DREAM's willingness to recompute schedules given new information, usually does not impact robustness. Individually optimizing for reduced computation and reduced communication leads to different ideal settings; while robustness can be gracefully traded for reduced communication, it is more difficult to trade robustness and computation. Most importantly, DREAM improves over DREA by gracefully trading lower communication and computation for lower robustness.

Our approach generalizes beyond the LP formulation of SREA in the generation of an SC schedule. The use of PSTN approaches such as (Fang, Yu, and Williams 2014) or (Santana et al. 2016) are amenable to these techniques. More generally, the techniques can be used to regenerate the optimal policy of an MDP provided new information.

While we can attempt to set threshold values to optimize communication or computation gains independently, the relative importance of computation and communication overhead are application dependent. If communication is more of a bottle-neck than computation, then setting a high m_{AR} and a high m_{SC} makes sense. If computing a new schedule is costly, e.g. when recomputing schedules on-board rather than at a remote controller base station, then lower m_{AR} and a low m_{SC} will provide a better execution strategy.

DREAM is a direct improvement over DREA in problems where remote agents have limited resources (such as in our UAV problem), or when computation is expensive. Cleverly choosing *when* to reschedule and communicate pre-

[2] While it may seem improvement occurs in the range $0.5 \leq m_{AR} \leq 1$ this is due to the graph package interpolation, not due to the actual raw data.

serves these resources. However, the current incarnation of DREAM allows agents to decide when to communicate at will. One area of future work is to adapt DREAM to handle communication *windows*. Regardless of whether this can be done entirely within the PSTN framework or not, preventing communications will force an agent to reason about robustness and the possible change in schedules when communications are possible; the resulting algorithm will produce less robust schedules than DREAM, but be able to reason about even more realistic communication limitations.

Another possible direction for future work is to test these algorithms in other kinds of simulations or applications. For instance, one could run tests in a time-based simulation, as opposed to an event-based simulation. It would be interesting to see how DREAM would perform in this context in comparison to DREA. Finally, testing with physics-based or real-world simulations currently considering the use of -aware PSTN techniques (e.g., Vaquero et al. (2019)) would yield results more directly relevant to practical situations. These further empirical explorations would create a stronger basis for justifying the value of SC and AR, and defining the trade-off between success rate and communication.

Acknowledgments

Funding for this work was graciously provided by the National Science Foundation under grant IIS-1651822, the NASA Advanced Exploration Systems (AES) program, and the Rose Hills Foundation. Thanks to the anonymous reviewers, HMC faculty, staff and fellow HEATlab members for their support and constructive feedback, with a special thanks to Brenner Ryan, DruAnn Thomas, and Zachary Dodds for their extra efforts in supporting this work.

References

Akmal, S.; Ammons, S.; Li, H.; and Boerkoel, J. C. 2019. Quantifying degrees of controllability in temporal networks with uncertainty. In *Proc. of the 29th International Conference on Automated Planning and Scheduling (ICAPS-19)*, To appear.

Brooks, J.; Reed, E.; Gruver, A.; and Boerkoel, J. C. 2015. Robustness in probabilistic temporal planning. In *Proc. of the 29th National Conference on Artificial Intelligence (AAAI-15)*, 3239–3246.

Cesare, K.; Skeele, R.; Yoo, S.; Zhang, Y.; and Hollinger, G. 2015. Multi-UAV exploration with limited communication and battery. In *Proc. of the IEEE Conference on Robotics and Automation (ICRA)*, 2230 – 2235.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 67(1-2):35–74.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. In *Knowledge Representation*, volume 49, 61–95.

Fang, C.; Yu, P.; and Williams, B. C. 2014. Chance-constrained probabilistic simple temporal problems. In *Proc. of the 28th National Conference on Artificial Intelligence (AAAI-16)*, 2264–2270.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 2002. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning* 49:193–208.

Li, B.; Jiang, Y.; Sun, J.; Cai, L.; and Wen, C.-Y. 2016. Development and testing of a two-UAV communication relay system. *Sensors* 16(10).

Lund, K.; Dietrich, S.; Chow, S.; and Boerkoel, J. 2017. Robust execution of probabilistic temporal plans. In *Proc. of the 31st National Conference on Artificial Intelligence (AAAI-17)*, 3597–3604.

Morris, P., and Muscettola, N. 2005. Temporal dynamic controllability revisited. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI-05)*, 1193–1198.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *Proc. of IJCAI-01*, 494–502.

Morris, P. 2014. Dynamic controllability and dispatchability relationships. In *Proc. of the International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR-14)*, 464–479.

Péret, L., and Garcia, F. 2004. On-line search for solving markov decision processes via heuristic sampling. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI-04)*, 530–534.

Quach, C.; Bole, B.; Hogge, E.; Vazquez, S.; Daigle, M.; Celaya, J.; Weber, A.; and Goebel, K. 2013. Battery charge depletion prediction on an electric aircraft. In *Proc. of the Annual Conference of the Prognostics and Health Management Society (PHM 2013))*.

Santana, P.; Vaquero, T.; Toledo, C.; Wang, A.; Fang, C.; and Williams, B. 2016. Paris: a polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In *Proc. of the 26th International Conference on Automated Planning and Scheduling (ICAPS-16)*, 267–275.

Tsamardinos, I. 2002. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Methods and Applications of Artificial Intelligence*, 97–108.

Vaquero, T.; Chien, S.; Agrawal, J.; Chi, W.; and Huntsberger, T. 2019. Temporal brittleness analysis of task networks for planetary rovers. In *Proc. 29th Int. Conf. on Automated Planning and Scheduling (ICAPS-2019)*, To appear.

Vidal, T., and Ghallab, M. 1996. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *Proc. of European Conference on Artificial Intelligence (ECAI-96)*, 48–54.

Vidal, T. 2000. Controllability characterization and checking in contingent temporal constraint networks. In *Proc. of the Principles of Knowledge Representation and Reasoning-International Conference (KR-00)*, 559–570.

Wu, F.; Zilberstein, S.; and Chen, X. 2011. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence* 175(2):487 – 511.