Journal of Artificial Intelligence Research 78 (2023) 849 - 900 Submitted 01/2023; published 11/2023

On Expected Value Strong Controllability

Niklas T. Lauffer NLAUFFER@BERKELEY.EDU
University of California, Berkeley
Berkeley, CA 94720

William B. Lassiter WLASSITER@GATECH.EDU
Georgia Institute of Technology
Atlanta, GA 30332

Jeremy D. Frank JEREMY.D.FRANK@NASA.GOV
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

The Probabilistic Simple Temporal Network with Uncertainty (PSTNU) is a variant of the Simple Temporal Network with Uncertainty (STNU) in which known probability distributions govern the timing of uncontrollable timepoints. Previous approaches to solving PSTNUs focus mininizing risk, that is, the probability of violating constraints. These approaches are not applicable in over-constrained controllability problems, when it is certain that all constraints can't be satisfied. We introduce the Weighted Probabilistic Simple Temporal Network with Uncertainty (WPSTNU), which extends the PSTNU by attaching a fixed value to the satisfaction of temporal constraints, and allows the schedule to violate some constraints in order to maximize the expected value of satisfying others. We study the problem of Expected Value Strong Controllability (EvSC) of WPSTNUs, which seeks a fixed-time schedule maximizing the expected value of satisfied constraints. We solve the EvSC problem using a mixed integer linear program (MILP) that bounds below the probability of satisfying constraints involving uncontrollable timepoints. While solving MILPs generally takes exponential time, we demonstrate our formulation's effective performance using scheduling problems derived from the HEATlab and MIT ROVERS data sets. We then show how to use this MILP to reschedule during execution, after time has passed and uncertainty is reduced. We describe different fixed-period rescheduling approaches, including time-based and event-based, and report on the most successful strategies compared to the expected value of the fixed schedule produced by the MILP. All of our methods are evaluated on problems with both symmetric and asymmetric (skewed) probability distributions. We show that periodically rescheduling improves the expected value when compared to the fixed schedule, and describe how the benchmark and skewness impact the schedule value improvement. The resulting analysis shows that solving EvSC problems on WPSTNUs is a viable alternative to solving over-constrained controllability problems.

1. Introduction

Since its introduction by (Vidal & Ghallab, 1996) and (Vidal & Fargier, 1999), there has been considerable research in the area of *controllability* of temporal networks in the presence of *uncertainty*. Controllability asks: can events be executed while satisfying temporal constraints in the presence of uncertain outcomes? Many previously studied solutions to this problem use the notion of controllability of Simple Temporal Networks under Uncertainty (STNUs) at their core. An STNU consists of controllable timepoints, representing events under the control of an agent; uncontrollable timepoints, representing events that occur at a bounded but unknown time after execution of controllable

timepoints; and simple temporal constraints between events that bound above or below their separation. The solutions to such problems are *strategies* to execute all events under the control of the agent that ensure no constraints are violated, regardless of the outcomes of uncertain events. The simplest strategy is a fixed execution time for each controllable timepoint. Not all STNUs can be controlled this way; in some cases, the strategy requires conditioning the time a controllable timepoints is executed in response to the occurrence of an uncontrollable timepoint. More complex problems combine temporal constraints, uncertainty, and preferences.

STNUs require all constraints to be satisfied regardless of the exact times when uncontrollable timepoints occur, and do not explicitly represent the probability distribution of uncontrollable timepoint occurrence. Uncertainty can be generalized so that the problem specifies *probability distributions* of event occurrence; these problems are referred to as Probabilistic Simple Temporal Networks (PSTNs). This leads to new *risk-minimization* and *chance-constrained* controllability problems. Minimizing risk means finding strategies that have minimum risk of any constraint violation. Chance constraints, on the other hand, allow some risk, as long as it does not exceed one (or possibly several) thresholds. Finally, some problems incorporate both uncontrollable and probabilistic constraints; these are referred to as PSTNUs.

The implicit assumption behind problems studied previously is that it is possible to schedule in such a way that the constraints are satisfied, and the risk is acceptable. What happens when it is *almost certain* that a constraint will be violated? A review of the recent literature shows that many PSTNU instances in current benchmarks are either not controllable, or are controllable but with high risk. For instance, consider the CAR-SHARING benchmark of (Fang, Yu, & Williams, 2014): only 184 of 1800 instances are strongly controllable, about 10%. For the ROVERS benchmark of (Santana, Vaquero, Toledo, Wang, & Williams, 2016), optimistically, 2840 of 4380 PSTNUs are strongly controllable, but in 911 of these cases, at least one probabilistic duration is squeezed to *a single value*. A more accurate assessment is that 1929 instances are strongly controllable, or about 44%. Finally, if we consider the HEATlab analyzed in (Lund, Dietrich, Chow, & Boerkoel, 2017), at least 80% of the 540 instances are strongly controllable [1].

Current approaches, particularly risk-bounding, do not adequately address the problem of uncontrollable PSTNUs. If the risk bound is too low, no strategy can be produced. If a risky strategy is produced, then an undesirable outcome at execution time is almost certain, and will violate a constraint, causing unexpected execution-time issues (e.g. 'freezing' execution, damaging the system, etc.). Some problem formulations allow constraints to be relaxed using a cost function on the relaxation, in addition to risk bounding. When the existing set of constraints cannot be satisfied to produce control strategies, relaxing some constraints up-front may ensure controllability, but as we will describe further below, these approaches have a variety of drawbacks. An alternative solution to such over-constrained problems is to let the execution strategy try to satisfy as many constraints as possible. If some constraints are more important than others, then a natural optimization criterion for the strategy is to maximize the *expected value* of satisfied constraints. This new, unexplored problem blends several notions explored in the controllability literature to date. Accepting risk implies accepting outcomes that violate some constraints. Applying preferences to satisfied constraints suggests control of expected schedule quality based on past information and the probability and cost of future constraint violations.

1. Of the 540 HEATlab instances, 436 were solved by at least one of the algorithms tested by (Abrahams, Chu, Diehl, Knittel, Lin, Lloyd, Boerkoel, & Frank, 2019) which is roughly 80%. However, the algorithms in (Abrahams et al., 2019) do not prove problems are *not SC*.

In this paper we extend PSTNUs to the *Weighted Probabilistic Simple Temporal Network with Uncertainty* (WPSTNU), and study the *Expected Value Strong Controllability* (EvSC) problem on WPSTNUs, that of finding a fixed schedule maximizing the expected value, and compare this problem to previous approaches. The outline of the paper is as follows.

We begin with preliminary notation and definitions in §2. We then provide motivation for the expected value formulation, illustrated by running examples in §3. We formally define the WPSTNU and the EvSC problem in §4. In §5 we describe our solution to the EvSC problem. We compute a strong lower bound for the probability that a given schedule satisfies a particular constraint on an uncontrollable timepoint, which we call 'at-risk' (AR) constraints. Computing these lower bounds for all AR constraints, in turn, yields a lower bound on the schedule's expected value that we can use as the objective value for a Mixed Integer Linear Program (MILP). Because the satisfaction of certain constraints may be essential to a plan's success, only a preordained subset of the constraints are considered for removal; the rest of the constraints must necessarily be satisfied. The value of these *rejectable* constraints must be traded against the expected value of satisfying AR requirement constraints.

In §6, we evaluate the performance of this formulation on benchmark instances derived from the HEATlab benchmark of (Lund et al., 2017) and the MIT ROVERS benchmark of (Santana et al., 2016). We construct new benchmarks from these instances by adding preferences to constraints, reducing the makespan, and adding more rejectable constraints to force tradeoffs among the constraints that are satisfied. We show that the MILP removes low-value constraints to improve the value of high-value AR constraints consistent with their value, and achieves close to optimal performance, despite the bounding approximation. While solving the MILP is exponential in the number of timepoints in the WPSTNU, we evaluate the empirical performance of Gurobi on the benchmark problems, and show most instances are solvable in under half a second; the hardest problems take under two minutes, which is comparable to previous results. We empirically quantify the error of the expected value produced by this MILP and show it is very small. Our benchmarks include both symmetric and skew probability distributions; to our knowledge, this is the first work to characterize the impact of skewness on any PSTNU variant.

In §7, we demonstrate the value of *rescheduling* periodically during execution, after time has passed and uncertainty is reduced, thereby improving the expected value. We define different fixed-period rescheduling approaches, including time-based and event-based, and formally prove that these rescheduling policies improve expected value compared to the fixed schedule produced by the MILP. For this study, we use both instances derived from the ROVERS benchmark as well as the HEATlab benchmark, and describe the difference in performance of rescheduling on these two benchmarks. We also compare the difference in rescheduling policy performance on symmetric and skew distributions for the ROVER benchmark, and describe the impact skewness has on improving schedule quality. We show that rescheduling is capable of improving the expected value by restoring as much as a third of the 'missing' value, that is, the difference between the total achievable value and the expected value of the fixed schedule for HEATlab instances, and 10% for ROVERS instances. We also show that problems with skewed probability distributions decrease the restored 'missing' value compared to the restoration of value achieved for problems with normal distributions.

Finally, in §8, we review of our findings on this new class of problems, and describe future work.

2. Notation and Definitions

In this section we will define controllability problems previously considered in the literature, as well as the notation we will use in this paper.

Definition 1 (STN). *(Dechter, Meiri, & Pearl, 1991) Simple Temporal Networks (STNs) consist of real-valued timepoints T with domain $t_i \in T = \mathbb{R}$. and constraints $c(t_i, t_j)$ of the form $(t_j - t_i) \in [l_{t_i,t_j}, u_{t_i,t_j}]$.*

Definition 2 (STNU). *(Vidal & Ghallab, 1996) (Vidal & Fargier, 1999) (Muscettola, Morris, & Vidal, 2001) Simple Temporal Networks with Uncertainty (STNUs) consist of:*

- *A set of* controllable *timepoints $A = \{a_1, a_2, \ldots a_k\}$, i.e. those assigned by the agent;*
- *A set of* uncontrollable *timepoints $R = \{r_1, r_2, \ldots r_\ell\}$ i.e. those assigned by the external world whose values $v(r_i)$ are unknown prior to execution;*
- *A set C of* requirement constraints *$c(t_i, t_j)$, where each $c(t_i, t_j)$ has the form $(t_j - t_i) \in [l_{t_i,t_j}, u_{t_i,t_j}]$;*
- *A set G of* contingent constraints*, where each $g(a_i, r_j)$ has the form $(r_j - a_i) \in [l_{a_i,r_j}, u_{a_i,r_j}]$ with $a_i \in A$, $r_j \in R$. We refer to a_i as the* activation timepoint *of the contingent constraint. Denote the* execution time *of a_i by $x(a_i)$. Uncontrollable timepoint r_j occurs at $v(r_j) = x(a_i) + \omega_j$, with $\omega_j \in \left[l_{a_i,r_j}, u_{a_i,r_j}\right]$ but ω_j, and thus $v(r_j)$, are only observed during execution.*

An STNU is a 4-tuple $<A,R,C,G>$. We denote the set of all timepoints as $T = A \cup R$; the domain of each $t_i \in T$ is $\mathbb{R}$.

STNUs include *only* contingent constraints on timepoints, without any probability distribution information available; hence the problem requires building a conformant plan that works regardless of the outcomes of all contingent links.

Definition 3 (Strong Controllability). *(Vidal & Fargier, 1999) Let U be an STNU. Let $\Omega = \times_{g(a_i,r_j)} [l_{a_i,r_j}, u_{a_i,r_j}]$ (the cross product of all possible outcomes of all contingent constraints). A schedule s is an assignment of times to $a_i \in A$. Denote the value of a_i in a schedule s by $s(a_i)$. U is* Strongly Controllable *(SC) if there is a schedule s such that for all realizations of uncertainty $\omega \in \Omega$, s satisfies all requirement constraints $c(t_i, t_j)$.*

To illustrate the ingredients of STNUs, consider a problem consisting of constraints over three timepoints, two controllable timepoints a_1 and a_2, and one uncontrollable timepoint r_1. We have one contingent constraint $g(a_1, r_1)$ with bounds $[1, 4]$, meaning r_1 occurs between 1 and 4 time units after a_1 executes. We have two requirement constraints, $c(a_1, a_2)$ with bounds $[0, 5]$, and $c(r_1, a_2)$ with bounds $[0, 3]$. Let schedule s assign $s(a_1) = 0$ and $s(a_2) = 4$. This assignment satisfies $c(a_1, a_2)$ since $4 - 1 \in [0, 5]$. We observe that $v(r_1) \in [1, 4]$. Finally, we observe that $4 - 1 = 3$ and $4 - 4 = 0$, proving *all* possible values $v(r_1)$ satisfy $c(r_1, a_2)$, i.e. $s(a_2) - v(r_1) \in [0, 3]$. Thus, our example STNU is *strongly controllable*.

We hereafter refer to requirement constraints between two controllable timepoints as the *Simple Temporal Network* (STN) constraints, denoted C_s, since they are the type of constraint found in

an STN, and those involving one or more uncontrollable timepoints as *At-Risk* (AR) constraints, denoted C_u. AR constraints are 'canonically' denoted $c(r_i, a_j)$, that is, $c(r_i, a_j)$ bounds $s(a_j) - v(r_i)$ to the range $[l_{r_i,a_j}, u_{r_i,a_j}]$. If $u_{r_i,a_j} > 0$, then a_j can only be executed after r_i; If $l_{r_i,a_j} < 0$, then a_j can be 'preemptively' executed before r_i occurs.

Modeling uncertainty using STNUs is rather restrictive, for STNUs do not incorporate information about the likelihood of the different outcomes. Suppose in the simple example above we know r_1 occurs on average 2.5 time units after a_1 with standard deviation of 0.5, distributed normally. STNUs can't express this knowledge, and thus it can't be used in generating schedules. Introducing continuous probabilities into controllability problems comes with some extra complexity; schedules may not account for all outcomes, and thus scheduling problems need to be extended with a notion of scheduling risk. These concepts are introduced in the following definitions.

Definition 4 (PSTN(U))**.** *(Tsamardinos, 2002), (Santana et al., 2016) Let V be a PSTNU. Let $<A,R,C,G,>$ be an STNU. A* probabilistic duration constraint *$d(a_i, r_j)$ has the form $r_j - a_i = \omega \in \Omega_{a_i,r_j}$ where $a_i \in A$, $r_j \in R$, and Ω_{a_i,r_j} is a random variable with probability density function P_{a_i,r_j}. Let D be a set of probabilistic duration constraints $d(a_i, r_j)$. A Probabilistic Simple Temporal Network (PSTN) is a 4-tuple $<A,R,C,D>$. As with STNUs, we refer to a_i as the activation timepoint of the duration constraint. A Probabilistic Simple Temporal Network with Uncertainty (PSTNU) is a 5-tuple $<A,R,C,G,D>$.*

In the sequel, we will assume w.l.o.g. that there is a bijection between probabilistic duration constraints and activation timepoints, so that for notational simplicity we can use a single index for the controllable activation timepoint, uncontrollable timepoint, and random variable involved in a probabilistic duration constraint, i.e. $d(a_i, r_i) \equiv r_i - a_i \in \Omega_i$, where Ω_i has density function P_i. Similarly, we will assume w.l.o.g. that there is a bijection between contingent links and uncontrollable timepoints, allowing the bounds of $g(a_i, r_i)$ to be denoted by $[l_{r_i}, u_{r_i}]$. With the introduction of PSTNUs, we note that AR constraints are now split into *probabilistic* AR constraints, involving uncontrollables in probabilistic duration constraints, and *uncertain* AR constraints, those involving uncontrollables in contingent links with no probability distribution information.

Risk, as introduced by (Fang et al., 2014), describes the probability that, given a schedule or strategy, an outcome $\omega \in \Omega$ violates one or more constraints. Typical approaches transform a PSTNU into an STNU by truncating the probability distributions to create bounded intervals, and then evaluate STNU controllability. To compute the risk of a solution to a PSTNU, we measure how much probability mass on each probabilistic duration is not covered after truncating the probability distributions' tails and throwing away the probability distribution, in order to transform it into a contingent link, i.e. transforming $d(a_i, r_i)$ to $g(a_i, r_i)$, in a manner similar to (Santana et al., 2016). A smaller interval for the contingent link is more likely to lead to an SC STNU, but incurs more risk. The definitions below bound above the risk, because our definition of PSTNU does not assume that the probabilities $P_i(\omega_i)$ are mutually independent.

Definition 5. *Let ρ_d:$D \rightarrow G$ transform a probabilistic duration constraint $d(a_i, r_i)$ into a contingent link $g(a_i, r_i)$ with bounds $[l_{r_i}, u_{r_i}] \subset \Omega_i$. Let $\rho_D = \cup d \in D(\rho_d)$, that is, ρ_D contains one 'squeezing' operation ρ_d for each probabilistic duration. Let V be a PSTNU. Then $\rho_D(V) = U$ where U is an STNU.*

Definition 6. *Let V be a PSTNU. Let $U = \rho_D(V)$ be an STNU derived from V. Let $d \in D$ and let $\rho_d(d(a_i, r_i)) = g(a_i, r_i)$. Let $[l_{r_i}, u_{r_i}]$ be the contingent constraint interval defined by $g(a_i, r_i)$. The* risk *of $d(a_i, r_i)$ relative to ρ_d, denoted $\delta(\rho_d(d(a_i, r_i)))$, is*

$$\delta(\rho_d(d(a_i, r_i))) = 1 - \int_{l_{r_i}}^{u_{r_i}} P_i(\omega).$$

The risk *of V relative to ρ_D, denoted $\delta(V, \rho_D)$, is bounded above by*

$$1 - \left(\prod_{d \in D} (1 - \delta(\rho_d(d(a_i, r_i)))) \right).$$

Definition 7. *A PSTNU V is SC with risk $\leq \Delta$ if there exists a mapping ρ_D such that $\delta(P, \rho_D) \leq \Delta$ and $U = \rho_D(V)$ is SC.*

Suppose we have a PSTNU V over three timepoints with a duration constraint $d(a_1, r_1)$, and two requirement constraints, $c(a_1, a_2)$ with bounds $[0, 5]$, and $c(r_1, a_2)$ with bounds $[0, 3]$. Suppose the probability P_1 associated with $d(a_1, r_1)$ is normally distributed with $\mu = 2.5, \sigma = 0.5$. Now suppose $U = \rho_D(V)$ truncates both tails of P_1, with contingent constraint $g(a_1, r_1)$ with bounds $[1, 4]$, recovering the strongly controllable STNU example above. The risk that the outcome $\omega = r_1 - a_1$ lies outside the bounds $1 \leq \omega \leq 4$ is 0.0027. The risk varies with both parameters of the normal. For example, increasing variance σ to 1 increases the risk to 0.13362. Similarly, moving the mean or modifying the distribution to be a skewed distribution will change the risk.

3. The Case For Expected Value

Prior to formally introducing the notion of a WPSTNU, we present a pair of examples in this section to motivate the viewing of PSTNs, and controllability problems, through the lens of expected value.

3.1 A Running Example: The Dust Devil

First, consider a planetary exploration rover with two imaging goals: one is a dynamic phenomenon of uncertain duration (a dust devil), and the second is a static target at the same location with a constraint relating to ideal lighting conditions. The dust devil image collection task spans 15 minutes, the subsequent image collection 10 minutes, and the drive to a position from which the dust devil and static target can be imaged takes 45 minutes. We assume the drive starts at time 0. The dust devil is expected to last between 40 and 70 minutes after the drive starts, ideally ending during the first imaging task. The ideal lighting window for the second image collection occurs between 60 and 70 minutes after starting the drive. The PSTN (there are no contingent links in this example) in Figure 1 depicts this scenario.

We first observe that the STNU obtained by transforming $d(a_0, r_0)$ into a contingent constraint $g(a_0, r_0) = [45, 60]$ is strongly controllable: equating a_0 with time 0, the assignment s with $s(a_1) = s(a_2) = 45$, $s(a_3) = s(a_4) = 60$, and $s(a_5) = 70$ satisfies all STN constraints, and satisfies AR constraint $c(r_0, a_3)$ for all outcomes $45 \leq v(r_0) \leq 60$. According to Definition 6, the risk associated with this transformation is simply the probability that $v(r_0) = s(a_0) + \omega_0$ falls outside of this prescribed range. Since there is only one AR constraint in the problem, for the purposes of this discussion, we denote the risk by $\delta(45)$ (parameterizing the risk δ with the start time of the projected contingent link) instead of $\delta(\rho_d(d(a_i, r_i)))$. Thus, the risk is given by

$$\delta(45) = 1 - \int_{45}^{60} P_0(\omega_0).$$

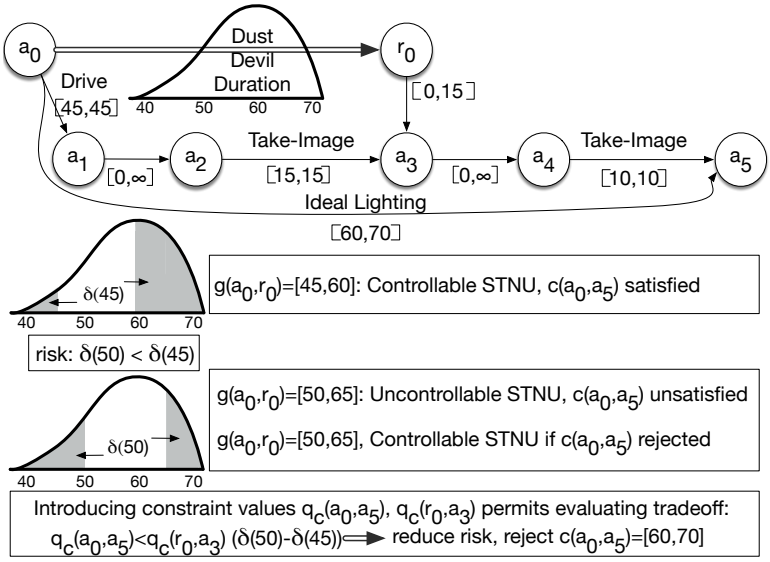


Figure 1: A controllability problem demonstrating the tradeoff of risk for sacrificing constraints. The controllability problem is shown at the top; arcs $a_i \Longrightarrow r_i$ are probabilistic durations, arcs $t_i \longrightarrow t_j$ are requirement constraints. The probability of satisfying $c(r_0, a_3)$ given two different assignments, $s(a_2) = 45$ and $s'(a_2) = 50$, is shown on the left. Sacrificing STN constraint $c(a_0, a_5)$ can increase expected value if AR constraint $c(r_0, a_3)$ has a high value, and the risk δ decreases sufficiently, as shown in the bottom box.

Suppose we instead convert $d(a_0, r_0)$ into $g(a_0, r_0) = [50, 65]$. The schedule s' with $s'(a_1) = 45$, $s'(a_2) = 50$, $s'(a_3) = 65$, $s'(a_4) = 65$, and $s'(a_5) = 75$ will assuredly fail to satisfy $c(a_0, a_5)$, but the risk of failing to satisfy $c(r_0, a_3)$ is now given by

$$\delta(50) = (1 - \int_{50}^{65} P_0(\omega_0)) < \delta(45).$$

The resulting STNU is not strongly controllable, but (as informally illustrated by the shaded region of the probability distribution $P_0(\omega_0)$ in Figure 1), we stand a better chance of our outcome ω_0 falling within the prescribed range.

Removing constraint $c(a_0, a_5)$ from the PSTN would result in a strongly controllable STNU with less risk. If maximizing the likelihood of imaging the dissipation of the dust devil is significantly more important to us than capturing the second image under ideal lighting conditions, it may be worth sacrificing the ideal lighting constraint in favor of an increased probability that we successfully image the end of the dust devil. PSTNUs focus on minimizing risk; all requirement constraints must be satisfied. By definition, there is no way to quantify the benefit of satisfying or rejecting requirement constraints in order to change the risk. Assigning a numerical value to each constraint $c(t_i, t_j)$ would allow us to evaluate the trade-offs between schedules that satisfy, or may satisfy, different sets of constraints. Constraints guaranteed to be satisfied by schedules contribute known value, while constraints that may be satisfied with some probability contribute *expected* value to a schedule. Only by doing this will we be able to evaluate whether or not it is better to reject the ideal lighting constraint and try to image the dust devil.

3.2 A Second Example: Trading Risk

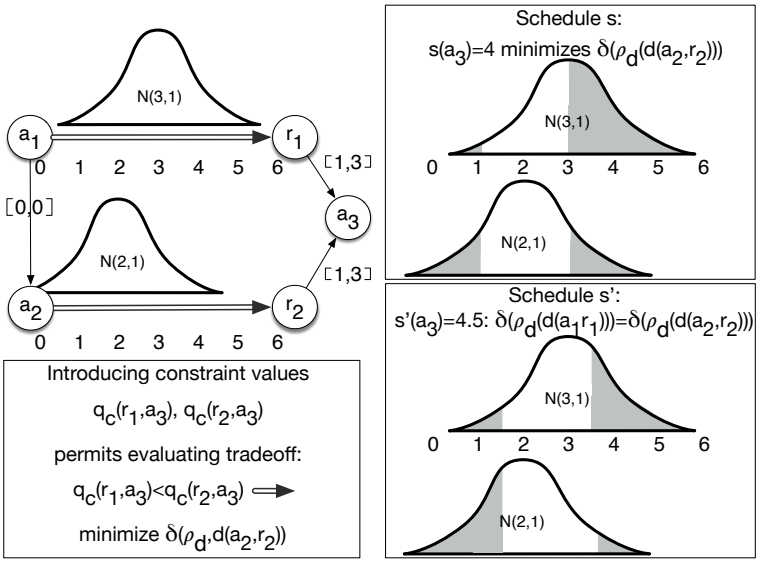


Figure 2: A controllability problem demonstrating the tradeoff of AR constraint risks. Schedule s chooses $s(a_3) = 4$ in order to maximize the probability of satisfying AR constraint $c(r_2, a_3)$. Schedule s' chooses $s'(a_3) = 4.5$ in order to minimize the risk. The concept of risk fails to properly encapsulate the trade-offs between s and s'. However, by introducing constraint valuations, they can be compared using expected value.

The previous example illustrates how sacrificing an STN constraint might increase the expected value of an AR constraint, improving the expected value for the problem as a whole. In order to more closely examine the sorts of problems in which a risk-bounding approach may not produce the most practical or effective schedule, we analyze a second scenario involving a tradeoff between risk for a pair of AR constraints. Consider a situation in which two probabilistic durations $d(a_1, r_1)$, $d(a_2, r_2)$ are constrained to start at the same time, and whose uncontrollable endpoints are constrained by a common end timepoint a_3 via constraints $c(r_1, a_3)$ and $c(r_2, a_3)$. The two uncertain durations are known to be normally distributed with $\mu = 2$ and 3 respectively, and unit variance. We examine two potential schedules for this scenario: a schedule s that attempts to maximize the probability of satisfying AR constraint $c(r_2, a_3)$ by scheduling a_3 at time $s(a_3) = 4$, and a schedule s' that minimizes risk by evenly distributing the probability of satisfying either at-risk constraint by scheduling a_3 at time $s'(a_3) = 4.5$. We now examine these two schedules in terms of risk. The risk for schedule s, given by

$$1 - (1 - \delta(\rho_d(d(a_1, r_1))))(1 - \delta(\rho_d(d(a_2, r_2)))) = 1 - (0.4773)(0.6827) = 0.6742,$$

is greater than that of s', given by

$$1 - (1 - \delta'(\rho_d(d(a_1, r_1))))(1 - \delta'(\rho_d(d(a_2, r_2)))) = 1 - (0.6247)(0.6247) = 0.6098.$$

However, if $c(r_2, a_3)$ is more valuable than $c(r_1, a_3)$, then schedule s may still be preferable. Risk-based approaches would only view schedule s as preferable to s' if constraint $c(r_1, a_3)$ was

relaxed or removed from the problem altogether. In the latter case, the risks of s and s' would simply be their probabilities of not satisfying $c(r_2, a_3)$, given by $1 - 0.6827 = 0.3173$ and $1 - 0.6247 = 0.3753$, respectively. Examples like this show that even for 'controllable' problems, i.e., those for which it is possible to satisfy all constraints, different valuations of constraints may lead to different schedules.

4. The WPSTNU

We now formalize the Weighted Probabilistic Simple Temporal Network with Uncertainty (WPSTNU) by adding constraint valuations $q_c(t_i, t_j)$, to a PSTNU, allowing us to define the Expected Value Strong Controllability (EvSC) problem on WPSTNUs. While a similar Dynamic Controllability problem can also be formalized, for the remainder of the paper, we will focus on Expected Value Strong Controllability. We revisit our examples and describe how they can be posed and solved as WPSTNUs. We then describe previous work on variants of PSTNUs and discuss why these variants differ from the WPSTNU.

4.1 WPSTNUs and EvSC: The Definitions

Definition 8 (WPSTNU)**.** *Let <A,R,C,G,D> be a PSTNU. We denote the set of AR constraints $c(r_i, a_j)$ by $C_u \subset C$ and the set of STN constraints $c(a_i, a_j)$ by $C_s \subset C$. We further denote the rejectable STN constraints by $C_r \subseteq C_s$. Let $q_c(t_i, t_j)$: $C_r \rightarrow \mathbb{R}^+$ and let Q be the set of all $q_c(t_i, t_j)$. A Weighted Probabilistic Simple Temporal Network (WPSTNU) is a 6-tuple <A,R,C,G,D,Q>.*

Intuitively, $q_c(t_i, t_j)$ is the value of satisfying constraint $c(t_i, t_j) \in C_r$ in the WPSTNU.

Definition 9. *Let W be a WPSTNU. Let S be the set of all schedules, and let $s \in S$ be a schedule satisfying all contingent constraints $g \in G$ and all non-rejectable STN constraints in $C_s \setminus C_r$. Let $\Omega = \times_i \Omega_i$ (the cross product of all possible outcomes of all probabilistic duration constraints) with joint density function P and let $\omega \in \Omega$ be a joint outcome of all probabilistic durations. Let*

$$\sigma(c(t_i, t_j), s, \omega) = \begin{cases} 1 & \text{if } c(t_i, t_j) \text{ is satisfied by } (s, \omega), \\ 0 & \text{otherwise,} \end{cases}$$

Then the value *of a schedule s given outcome $\omega \in \Omega$ is*

$$f(s, \omega) = \sum_{c \in C} q_c(t_i, t_j)\ \sigma(c(t_i, t_j), s, \omega),$$

and the expected value of s is

$$g(s) = \int_{\omega \in \Omega} f(s, \omega) P(\omega).$$

The Expected Value Strong Controllability *(EvSC) problem defined over WPSTNU U is to find $s \in S$ satisfying all contingent constraints and maximizing $g(s)$.*

In the definition above, σ determines whether a constraint $c(t_i, t_j)$ is satisfied by schedule s and outcome ω. Satisfied constraints $c(t_i, t_j)$ accrue value $q_c(t_i, t_j)$ in $f(s, \omega)$. The expected value of a schedule incorporates the probability of outcomes that influence the satisfaction of constraints on uncontrollable timepoints, e.g., $q_c(r_i, a_j)$. We will occasionally abuse notation and write

$\sigma(c(a_i, a_j), s)$ when σ is used to determine satisfaction of an STN constraint, and we don't need to refer to ω.

The risk minimization setting assumes that violation of any single constraint would lead to failure in the schedule execution. By contrast, the EvSC problem inherently assumes that violation of certain constraints, while not desirable, would not disturb the execution of the remainder of the schedule. For a fixed-time schedule s, an STN constraint in C_s either is satisfied and provides full value, or rejected and provides no value. By contrast, an AR constraint $c(r_i, a_j)$ only has expected value prior to execution time. A disproportionately large assigned value $q_c(t_i, t_j)$ incentivizes satisfaction of STN constraints, or ensures a high probability of satisfying AR constraints. STN constraints that are critical to the integrity of the whole schedule (such as a drive for a rover) can be made non-rejectable.

4.2 Revisiting the Examples

We now look deeper at the fundamental tradeoff in EvSC: sacrificing a constraint to improve the overall expected value of a schedule. In Figure 1, there is only AR constraint, namely $c(r_0, a_3)$. As above, assume $s(a_0) = s'(a_0) = 0$, and let s be a schedule in which $s(a_2) = 45$ and s' be a schedule in which $s'(a_2) = 50$. As noted previously, s satisfies all the STN constraints, but the range of outcomes satisfying $c(r_0, a_3)$ is $45 < v(r_0) \leq 60$, which has low probability. Schedule s' violates a single constraint, namely, $c(a_0, a_5)$, but changes the range of outcomes satisfying $c(r_0, a_3)$ to $50 < v(r_1) \leq 65$, increasing the probability. Committing to a schedule up-front that violates $c(a_0, a_5)$ lets us increase the probability $c(r_0, a_3)$ is satisfied, potentially increasing the expected value of the schedule. In order to make violating $c(a_0, a_5)$ maximize the expected value, we would need the *relative* values of $q_c(r_0, a_3)$ and $q_c(a_0, a_5)$ to satisfy the inequality:

$$\left(q_c(r_0, a_3) \int_{45}^{60} P_0(\omega_0)\right) + q_c(a_0, a_5) < \left(q_c(r_0, a_3) \int_{50}^{65} P_0(\omega_0)\right)$$
$$\iff q_c(a_0, a_5) < q_c(r_0, a_3) \left(\int_{50}^{65} P_0(\omega_0) - \int_{45}^{60} P_0(\omega_0)\right).$$

Denote the difference in risk by

$$\Delta = \left(\int_{50}^{65} P_0(\omega_0) - \int_{45}^{60} P_0(\omega_0)\right),$$

Then we have

$$q_c(a_0, a_5) < \Delta q_c(r_0, a_3).$$

If Δ is 'small', the inequality above is only satisfied if $q_c(r_0, a_3)$ is 'large'. Put another way, $q_c(r_0, a_3)$ needs to be at least a factor of $\frac{1}{\Delta}$ larger than $q_c(a_0, a_5)$ in order to justify violating $c(a_0, a_5)$.

The example in Figure 2 shows that PSTNUs cannot distinguish between multiple sources of risk. Approaching this problem from the perspective of expected value allows us to consider this trade-off in a more nuanced way. If we prefer satisfying constraint $c(r_2, a_3)$ without altogether

(implicitly) throwing out $c(r_1, a_3)$, we could assign $q_c(r_2, a_3) = 3$ and $q_c(r_1, a_3) = 1$, respectively, so that the expected value of schedule s, given by

$$g(s) = 3(0.6827) + 1(0.4773) = 2.5253,$$

would make it preferable to schedule s' with expected value

$$g(s') = 3(0.6247) + 1(0.6247) = 2.4986.$$

4.3 Previous Work

In our first example, we would like to evaluate the trading of satisfaction of the lighting constraint $c(a_0, a_5)$ with satisfaction of the risky dust devil observation constraint, $c(r_0, a_3)$. The right strategy depends on the relative importance of satisfying $c(r_0, a_3)$ and $c(a_0, a_5)$, *and* the probability of satisfying $c(r_0, a_3)$, which requires formulating the expected value of a schedule or strategy. In our second example, the addition of value on constraints transforms the tradeoff of risk, which might otherwise not favor one AR constraint over another, into an expected value problem. As we now explain, these types of tradeoffs are not easily captured by previous work on overconstrained PSTNUs, due to either the lack of constraint value or the way constraint values are used; nor is it easily captured by previously developed formalisms employing both constraints and preferences, because they lack explicit representation of probability.

As noted in the definitions in §2, PSTNs were originally introduced by (Tsamardinos, 2002), with empirical studies of different algorithms performed in (Santana et al., 2016), and (Brooks, Reed, Gruver, & Boerkoel, 2015). The chance-constrained PSTN (called cc-pSTPs in (Fang et al., 2014)) constrains risk instead of minimizing it. A single constraint on risk is introduced in (Fang et al., 2014); (Wang & Williams, 2015) introduces a cc-pSTP variant in which there can be many risk constraints. When the risk bounds can't be satisfied, the resulting problem instance is deemed unsolvable.

One previously explored approach for over-constrained PSTNs is to search over *relaxations* for a problem that can be transformed into a controllable STNU with some bounded risk; (Yu, Fang, & Williams, 2015; Yu, Williams, Fang, Cui, & Haslum, 2017) use this approach for over-constrained cc-pSTPs. The search is guided by the *costs* of relaxations of either the requirement constraints or the chance constraint. Each relaxation has a cost; the approach involves finding a minimum-cost relaxation that leads to controllability of the underlying STNU. The WPSTNU is more limited than than relaxable cc-pSTP, in that only rejection of requirement constraints is permitted, and constraints have scalar value instead of more general relaxation costs. In Figure 1, a relaxed constraint $c(a_0, a_5) = [60, 75]$ (not shown) would lead to a controllable STNU with $s(a_2) = 55$. Applying the notion of relaxable cc-PSTPs to Figure 2, one could either permit higher risk or widen the bounds of the constraints, and impose a larger cost for relaxation of (say) $c(r_1, a_3)$ than $c(r_1, a_3)$. However, the cc-PTSP requires both (relaxed) constraints, including the risk bound, to be satisfied at scheduling time. As we saw in Figure 1, the WPSTNU permits the expression of the expected value proposition by combining the scalar benefit of satisfying constraints with the calculated probability of success. We also observe that there is no easy way to capture the tradeoff in expected value using relaxation costs. To see why, observe that all risk is treated equally by the relaxable cc-PSTP approach; further, the relaxation costs to fix violations and risk allocation are treated separately by the solver. Thus, the WPSTNU complements the relaxable cc-pSTP approach to handling over-constrained PSTNs.

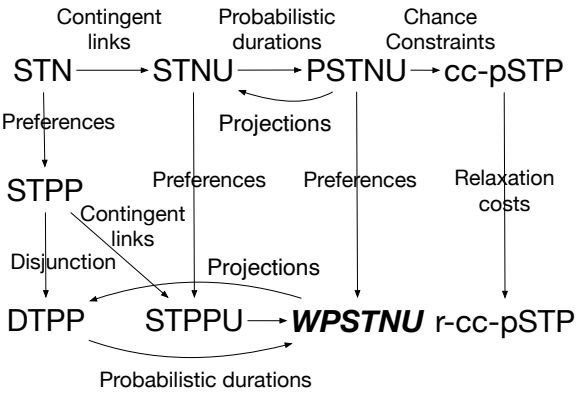


Figure 3: The relationship between previous approaches to over-constrained controllability problems and WPSTNUs.

A related approach is the Controllable Conditional Temporal Problem with Uncertainty (CCTPU) of (Yu et al., 2015), which combines STNUs (without probabilities) with costs of relaxations, rewards for assignments, and conditional generation of events and constraints based on non-temporal finite-domain variables. WPSTNUs are similar to the CCTPU, in that we can choose which constraints to satisfy. WPSTNUs are more limited than CCTPUs in that every timepoint of a WPSTNU must be scheduled, relaxations are limited to the rejection of constraints, and WPSTNUs only represent expected value, not a combined cost-benefits analysis. WPSTNUs are more general than CCTPUs in that they include probabilities, not just qualitative uncertainty of uncontrollable events.

The approaches above use relaxation *costs* as a means to enable controllability and satisfaction of risk bounds. Alternatively, as we do with the WPSTNU, the problem can be posed by using *preferences* on 'degree of satisfaction' of STN constraints, which could then be traded against the expected value of satisfying AR constraints. This could be done using simple semi-convex preference functions, combined with 'min', to achieve tractability, as is done with the Simple Temporal Problem with Preferences and Uncerainty (STPPU) (Rossi, Venable, & Yorke-Smith, 2006). However, the assumptions that preserve tractability are too limiting; in particular, the 'min' function will report the worst preference achieved for any constraint, which could be 0 (representing a 'violated' constraint). WPSTNUs are a variant of the Disjunctive Temporal Problem with Preferences (DTPP) (Peinter, Moffitt, & Pollack, 2005), in which not all constraints can be satisfied, and the best set of constraints to satisfy must be found by search. The WPSTNU is a strict generalization of the DTPP; while the value of satisfying $c(a_i, a_j)$ is captured by $q_c(a_i, a_j)$, the expected value of satisfying $c(r_i, a_j)$ is a nontrivial function of timepoint assignments, rather than a constant associated with the disjunctive decisions. Each requirement constraint can be expressed as a disjunction where satisfying the 'trivial' constraint has zero value and satisfying the original constraint has value $q_c(t_i, t_j)$.

The expected value formulation is common in Markov Decision Processes (MDPs) and their numerous variants. While temporal MDPs have been considered, the continuous time nature of the state space precludes using formulations such as time-dependent MDPs (Boyan & Littman, 2000); the desire to express state spaces representing violated constraints makes other time-based MDP approaches, e.g. (Weld & Mausam, 2006), inappropriate.

Figure 3 describes the relationship between previous approaches to over-constrained problems on temporal networks with uncertainty, and the newly defined WPSTNU.

5. Solving the EvSC Problem

We now describe how we will solve the EvSC problem on WPSTNUs. Central to EvSC is the ability to trade (possibly expected) value between constraints. As with previous approaches to PSTNs, the WPSTNU can be formulated as a nonlinear optimization problem:

$$\max_s g(s) = \max_s \int_{\omega \in \Omega} \left(\sum_{c \in C_s} \sigma(c(a_i, a_j), s, \omega) q_c(r_i, a_j) + \sum_{c \in C_u} \sigma(c(r_i, a_j)s, \omega) q_c(r_i, a_j) \right) P(\omega). \tag{1}$$

Since the value of $\sigma(c(a_i, a_j)s, \omega)$ only depends on s and not ω, this simplifies to:

$$\max_s \left[\sum_{c \in C_s} \sigma(c(a_i, a_j)s) q_c(a_i, a_j) + \int_{\omega \in \Omega} \left(\sum_{c \in C_u} \sigma(c(r_i, a_j)s, \omega) q_c(r_i, a_j) \right) P(\omega) \right]. \tag{2}$$

(Fang et al., 2014) and (Wang & Williams, 2015) use nonlinear solvers to handle the allocation of risk. Such solvers are often slow and prone to numerical instability. Inspired by previous MILP formulations ((Santana et al., 2016) and (Lund et al., 2017)), we will solve the EvSC problem by building a MILP whose solution is a schedule that is guaranteed to bound below the expected value of the optimal schedule. We first show how to process the probability distributions to build a piecewise linear bound on the probabilities of satisfying AR constraints. We then describe the rest of the MILP, and provide a bound on the MILP quality.

5.1 Bounding The Probability of Satisfaction

Clearly if there is no schedule satisfying all non-rejectable constraints in $C_s \setminus C_r$ then maximizing the expected value is pointless, so we assume there is such a schedule (which can be checked easily). Solving an EvSC problem must strike a balance between satisfying the STN constraints in C_r (thus surely obtaining their value) and maximizing the chances of satisfying high-value (but uncertain) AR constraints. An optimal schedule assigns timepoints to maximize the value we expect to obtain from the constraints in C_u, and the guaranteed value of the satisfied subset $C'_r \subseteq C_r$ of rejectable constraints.

We need to explicitly represent the probability that an uncontrollable event's actual time of occurrence, which is $v(r_i) = s(a_i) + \omega_i,\ \omega_i \in \Omega_i$, leads to a violation of AR constraint $c(r_i, a_j)$ with bounds $[l_{r_i,a_j}, u_{r_i,a_j}]$. This means we can't directly use $\delta(\rho_d(d(a_i, r_i)))$ from Definition 6, because this definition only computes the risk by measuring how much of the probability mass of $d(a_i, r_i)$ is covered by $g(a_i, r_i) = \rho_d(d(a_i, r_i))$. When computing the expected values, it is helpful to imagine a triangle consisting of a (possibly implied) STN constraint $c(a_i, a_j)$, a probabilistic duration constraint $d(a_i, r_i)$, and an AR constraint $c(r_i, a_j)$ as shown in Figure 4. Given a schedule s, regardless of whether or not s satisfies $c(a_i, a_j)$, outcome $v(r_i) = s(a_i) + \omega_i$ satisfies the constraint $c(r_i, a_j)$ if

$$l_{r_i,a_j} \leq s(a_j) - v(r_i) \leq u_{r_i,a_j}.$$

After substitution,

$$l_{r_i,a_j} \leq s(a_j) - s(a_i) - \omega_i \leq u_{r_i,a_j}.$$

Because $v(r_i) = s(a_i) + \omega_i$, the later $v(r_i)$ occurs, the smaller $s(a_j) - s(a_i) - \omega_i$ is, and vice versa, the earlier $v(r_i)$ occurs, the larger $s(a_j) - s(a_i) - \omega_i$ becomes. Isolating (positive) ω_i inside the

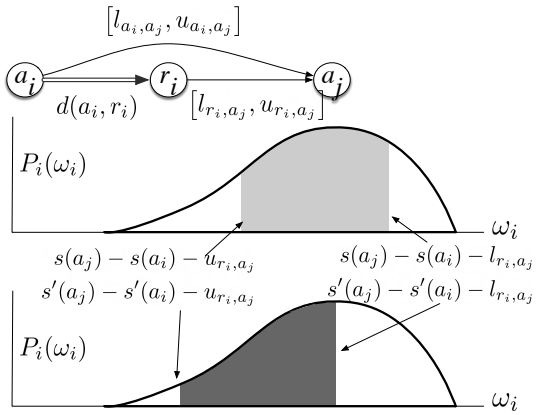


Figure 4: Computing expected value for an AR constraint. The 'triangle' involving a_i, r_i and a_j is formed by a duration constraint $d(a_i, r_i)$, an STN constraint $c(a_i, a_j)$, and an AR constraint $c(r_i, a_j)$. Assignments to controllable timepoints $s(a_i)$ and $s(a_j)$ may or may not satisfy STN constraint $c(a_i, a_j)$ (top). The range of probability covered is defined by the bounds on the AR constraint,$[l_{r_i,a_j}, u_{r_i,a_j}]$, and the temporal distance between assignments $s(a_j) - s(a_i)$. The first assignment s (middle) positions the AR constraint range over the mode of the probability density function, while the second assignment s' (bottom) pushes the range away from the mode due to a smaller distance $s'(a_j) - s'(a_i)$.

inequalities yields:

$$s(a_j) - s(a_i) - u_{r_i,a_j} \leq \omega_i \leq s(a_j) - s(a_i) - l_{r_i,a_j}.$$

Given schedule s and thus $s(a_i)$ and $s(a_j)$, the probability of satisfying the constraint $c(r_i, a_j)$, and obtaining value $q_c(r_i, a_j)$, is

$$p_{ij}(s) = \int_{s(a_j)-s(a_i)-u_{r_i,a_j}}^{s(a_j)-s(a_i)-l_{r_i,a_j}} P_i(\omega_i).$$

The probability of satisfying AR constraint $c(r_i, a_j)$ depends on where the range of ω_i satisfying the constraint, namely $[s(a_j) - s(a_i) - u_{r_i,a_j}, s(a_j) - s(a_i) - l_{r_i,a_j}]$, falls on the probability distribution $P_i(\omega_i)$. An outcome may be unlucky, either violating the lower bound l_{r_i,a_j} because ω_i is too large, or violating the upper bound u_{r_i,a_j} because ω_i is too small. The *size* of this range of outcomes is defined by the AR constraint bounds, $[l_{r_i,a_j}, u_{r_i,a_j}]$, which is independent of the actual schedule. However, the difference $s(a_j) - s(a_i)$, which is a function of the schedule, determines *what portion* of $P_i(\omega_i)$ is covered, as shown in Figure 4. If the mode of $P_i(\omega_i)$ is in the center of this interval, then probability $p_{ij}(s)$ will be high or maximal; if the mode is near one side or the other, or outside the interval, the probability will be low.

In general, we must search over many possible schedules, meaning $s(a_j) - s(a_i)$ will vary, and so will the value of the integral defining the probability of success. For a probabilistic duration $d(a_i, r_i)$ and AR constraint $c(r_i, a_j)$ the function

$$F_{ij}(x) = \int_{x-u_{r_i,a_j}}^{x-l_{r_i,a_j}} P_i(\omega_i),$$

represents *all* possible probabilities of not satisfying $c(r_i, a_j)$ that might arise as a result of the choices of $s(a_i), s(a_j)$ during search. $F_{ij}(x)$ has as its only 'free' variable the distance $x = a_j - a_i$; the bounds from $c(r_i, a_j)$ are constant parameters. If $P_i(\omega_i)$ is unimodal, then $F_{ij}(x)$ is unimodal but not necessarily concave.

We note that the term

$$\int_{\omega \in \Omega} \left(\sum_{c \in C_u} \sigma(c(r_i, a_j), s, \omega) q_c(r_i, a_j) \right) P(\omega)$$

from Equation 2 can be written

$$\int_{\omega \in \Omega} \left(\sum_{c \in C_u} \sigma(c(r_i, a_j), s, \omega) q_c(r_i, a_j) P(\omega_i) \right)$$

because the probability of obtaining $q_c(r_i, a_j)$ depends only on ω_i. But as we just showed, the bounds of integration cover all of the outcomes of $P_i(\omega_i)$ for which $\sigma(c(r_i, a_j), s, \omega) = 1$. That means this term can now be written

$$\sum_{c \in C_u} \left(\int_{s(a_j) - s(a_i) - u_{r_i, a_j}}^{s(a_j) - s(a_i) - l_{r_i, a_j}} q_c(r_i, a_j) P_i(\omega_i) \right)$$

The above term, in turn, can be expressed using our newly defined function $F_{ij}(x)$:

$$\sum_{c \in C_u} F_{ij}(s(a_j) - s(a_i)) q_c(r_i, a_j)$$

which means we can rewrite the expected value formula in Equation 2 as:

$$g(s) = \sum_{c \in C_s} \sigma(c(a_i, a_j), s) q_c(a_i, a_j) + \sum_{c \in C_u} F_{ij}(s(a_j) - s(a_i)) q_c(r_i, a_j). \tag{3}$$

In the sequel, we assume that P_i follows a normal, uniform, or beta (with $\alpha, \beta \geq 1$) distribution, although the formulation may hold for other unimodal distributions with concavity about their mode. We would like to solve the maximum expected value problem using a MILP. Given the non-linearity of $F_{ij}(x)$, we must use a linear approximation of $F_{ij}(x)$ in order to use a MILP solver. We choose to approximate from below to obtain a conservative estimate on the probability of constraint satisfaction, and by extension, expected value. While $F_{ij}(x)$ is not concave over its entire range, we can bound its concave region (including its mode) from below with a series of y_{ij} linear inequalities defined by functions $m_{ij}^k x + c_{ij}^k,\ k = 1, \ldots, y_{ij}$ that underestimate the probability of obtaining value $q_c(r_i, a_j)$, and from the left and right with appropriately chosen values $\underline{\alpha}_{ij}$ and $\overline{\alpha}_{ij}$.

The construction of this piecewise linear approximation, shown in Figure 5, is accomplished as follows. We first evaluate $F_{ij}(x)$ at a given number of equally spaced x-values surrounding its mode (the more evaluations, the better the approximation) and draw line segments between consecutive points. Then, starting from the left and moving right, we choose the left endpoint of the first line segment whose slope is not larger than that of its predecessor, and, starting from the right and moving left, we choose the right endpoint of the first line segment whose slope is not smaller than that of its predecessor. These points will serve as our approximate left and right inflection points,

and we use the x-intercepts of the lines tangent to F_{ij} at these points as our $\underline{\alpha}_{ij}$ and $\overline{\alpha}_{ij}$ values. Furthermore, the equations of the lines defining the segments found between these points together with those two tangent lines will serve as our set of bounding functions $m_{ij}^k x + c_{ij}^k$. With a fine enough discretization, these functions collectively define a good lower approximation of the concave part of F_{ij}.

We must approximate $|C_u|$ such functions F_{ij}, one per AR constraint. We must perform integrals to construct each of their approximations. Once we have decided how many points each piecewise linear approximation will have, however, we can limit the number of integrations. We also have the benefit of doing definite integrals, which is often easy for 'nice' distribution families (e.g., normal distributions). Finally, our approach handles all the integrations offline as pre-processing steps prior to solving the MILP, instead of online, which is the approach taken by (Fang et al., 2014).

5.2 A MILP for EvSC

Our MILP for EvSC is shown in Figure 6. There are $|A|$ continuous variables a_i representing scheduled times $s(a_i)$ for controllable timepoints, $|C_u|$ continuous variables λ_{ij} representing a lower bound on the probability of satisfying AR constraint $c(r_i, a_j)$, $|C_u|$ binary variables ξ_{ij} used to track whether or not $a_j - a_i$ falls within the concave portion of F_{ij}, and $|C_r|$ binary variables β_{ij} used to track whether or not the schedule satisfies each rejectable constraint. The constants (indicated in bold) are constraint values $q_c(t_i, t_j)$, requirement constraint bounds $[l_{t_i,t_j}, u_{t_i,t_j}]$, M, $\underline{\alpha}_{ij}$ and $\overline{\alpha}_{ij}$, and the slopes m_{ij}^k and intercepts c_{ij}^k of the piecewise linear approximations of functions F_{ij}.

Constraints (1) ground the schedule's origin point a_0 at time 0, and constraints (2) and (3) enforce non-rejectable STN constraints. For each STN constraint in C_r, the binary variable β_{ij} is used in the 'big-M' linear constraints (4) and (5); the constant $M > 0$ is sufficiently large to render (4) and (5) non-binding when $\beta_{ij} = 0$ (i.e. the STN constraint is 'turned off'). Linear constraints (6) and (7) function in much the same way for each AR constraint in C_u; the binary variable ξ_{ij} may only be set to 1 if $a_j - a_i$ falls within the 'good' (i.e. concave) region of F_{ij}. When $a_j - a_i$ falls outside of the good region, λ_{ij} is forced to 0 by its upper bound ξ_{ij} with linear constraint (8). Linear Constraints (9) bound λ_{ij} above by our approximation for F_{ij}; the big-M term must also be included to make these constraints non-binding when $a_j - a_i$ falls outside of the good region, since

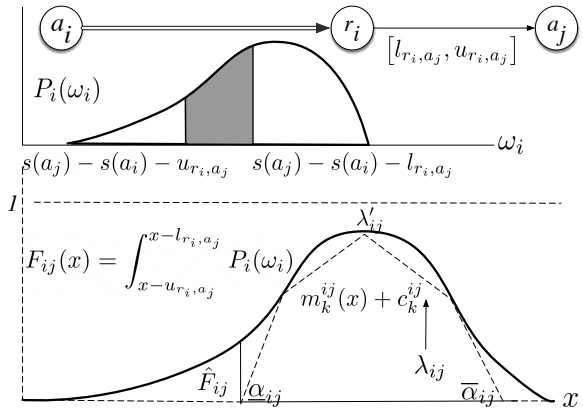


Figure 5: Bounding the probability to construct a MILP.

$$\max \sum_{C_u} \lambda_{ij} \mathbf{q_c}(\mathbf{r_i}, \mathbf{a_j}) + \sum_{C_r} \beta_{ij} \mathbf{q_c}(\mathbf{a_i}, \mathbf{a_j})$$

$$\text{s.t. } a_0 = 0 \tag{1}$$

$$a_j - a_i \leq \mathbf{u_{a_i,a_j}} \qquad \forall c(a_i, a_j) \in C_s \setminus C_r \tag{2}$$

$$a_j - a_i \geq \mathbf{l_{a_i,a_j}} \qquad \forall c(a_i, a_j) \in C_s \setminus C_r \tag{3}$$

$$a_j - a_i \leq \mathbf{u_{a_i,a_j}} + \mathbf{M}(1 - \beta_{ij}) \qquad \forall c(a_i, a_j) \in C_r \tag{4}$$

$$a_j - a_i \geq \mathbf{l_{a_i,a_j}} - \mathbf{M}(1 - \beta_{ij}) \qquad \forall c(a_i, a_j) \in C_r \tag{5}$$

$$a_j - a_i \leq \overline{\boldsymbol{\alpha}}_{ij} + \mathbf{M}(1 - \xi_{ij}) \qquad \forall c(r_i, a_j) \in C_u \tag{6}$$

$$a_j - a_i \geq \underline{\boldsymbol{\alpha}}_{ij} - \mathbf{M}(1 - \xi_{ij}) \qquad \forall c(r_i, a_j) \in C_u \tag{7}$$

$$\lambda_{ij} \leq \xi_{ij} \qquad \forall c(r_i, a_j) \in C_u \tag{8}$$

$$\lambda_{ij} \leq \mathbf{m}_{ij}^k (a_j - a_i) + \mathbf{c}_{ij}^k + \mathbf{M}(1 - \xi_{ij}) \qquad \forall c(r_i, a_j) \in C_u$$

$$\forall k = 1, \ldots, y_{ij} \tag{9}$$

$$\beta_{ij} \in \{0, 1\} \qquad \forall c(a_i, a_j) \in C_r \tag{10}$$

$$\xi_{ij} \in \{0, 1\} \qquad \forall c(r_i, a_j) \in C_u \tag{11}$$

Figure 6: MILP maximizing the expected value of a consistent WPSTNU.

otherwise λ_{ij} would potentially be forced below 0, degrading the accuracy of our expected value estimate.

Finally, we explain how the objective function bounds below the schedule expected value. Given a rejectable STN constraint $c(a_i, a_j)$, we obtain value $q_c(a_i, a_j)$ whenever $a_j - a_i$ falls within its bounds (and thus $\beta_{ij} = 1$); given an AR constraint $c(r_i, a_j)$, we expect to receive value $q_c(r_i, a_j)F_{ij}(a_j - a_i)$. Since, by construction of the MILP, λ_{ij} is a lower bound on $F_{ij}(a_j - a_i)$, the MILP objective value will be a lower bound on the true expected value. Posing the objective function in this way highlights the trade-off between satisfying rejectable STN constraints and ensuring high probabilities of satisfaction for AR constraints.

5.3 Quality of the MILP Solution

As described in the previous section, the piecewise linear functions in constraints (9) and the binary variables ξ_{ij} together serve to bound the true probability of the schedule satisfying a given AR constraint from below. Thus, the optimal solution to the MILP will generally be lower than the true expected value. Note that error in the approximation will generally occur below $\underline{\alpha}_{ij}$ or above $\overline{\alpha}_{ij}$, but it could also occur near the mode of $F_{ij}(x)$ if it is approximated by only a small number of pieces. To see that nontrivial error in solutions can occur, we observe that if the piecewise linear approximation is flat (slope 0) near the maximum of $F_{ij}(x)$, leading to the same probability λ_{ij} in many MILP solutions, the true curve $F_{ij}(x)$ will yield better solutions only found by nonlinear optimization starting at the arbitrary solution found using the piecewise linear approximation. Let

$$\hat{G}_{ij} = \max_x \{F_{ij}(x) - (m_{ij}^k x + c_{ij}^k)\},$$

be the largest difference between the true value of $F_{ij}(x)$ and its piecewise linear lower bound. The worst-case error of any solution to a consistent WPSTNU is no more than

$$\hat{\epsilon} = \sum_{C_u} q_c(r_i, a_j)\hat{G}_{ij}.$$

Given a schedule s, let λ_{ij}^s be the approximated probabilities of success for the schedule from the MILP solution. The error $\hat{\epsilon}(s)$ of this schedule is

$$\hat{\epsilon}(s) = \sum_{C_u} q_c(r_i, a_j) \left(F_{ij} \left(s(a_j) - s(a_i) \right) - \lambda_{ij}^s \right),$$

and trivially $\hat{\epsilon}(s) \leq \hat{\epsilon}$. Suppose that the true optimal schedule is m with value M, and let $\hat{M} = M - \hat{\epsilon}(m)$, that is, the objective value of schedule m according to the MILP. Suppose b is an optimal schedule according to the MILP with objective value $\hat{B}$. We thus have the following lemma:

Lemma 1. *Given a WPSTNU whose true optimal schedule m has expected value M and whose maximum MILP formulation error is $\hat{\epsilon}$, let b be an optimal solution to the MILP formulation with objective value $\hat{B}$. Then $\hat{B} \leq M \leq \hat{B} + \hat{\epsilon}$.*

Proof. Since each λ_{ij} value is bounded above by F_{ij}, we have $\hat{B} \leq M$. Since b is the optimal schedule according to the MILP, $\hat{M} \leq \hat{B}$. We also know $M = \hat{M} + \hat{\epsilon}(m)$ and thus, $M \leq \hat{B} + \hat{\epsilon}(m)$. Finally, $\hat{\epsilon}(m) \leq \hat{\epsilon}$, so $M \leq \hat{B} + \hat{\epsilon}$. □

Computing the worst-case error $\hat{\epsilon}$ for a schedule requires computing $\hat{G}_{ij}$ for all AR constraints $c(r_i, a_j)$. The piecewise linear approximation inside the concave region can be made good enough that the error $F_{ij}(x) - (m_{ij}^k x + c_{ij}^k)$ is maximized at the tails, and therefore we only need to compute the difference at $\underline{\alpha}_{ij}$ and $\overline{\alpha}_{ij}$ to find $\hat{G}_{ij}$.

5.4 Complexity, Soundness, and Incompleteness

The number of variables and constraints employed by a MILP formulation is often used as indicators of how hard it is to solve. Our MILP involves $|A| + |C_u|$ continuous and $|C_u| + |C_r|$ binary variables. In terms of constraints, it has $2|C_s|$ of them enforcing the lower and upper bounds of STN constraints, $2|C_u|$ constraints checking whether or not $a_j - a_i$ values fall within their 'good' regions, $|C_u|$ constraints bounding λ values from above within their 'bad' regions, and no more than $y_*|C_u|$ constraints bounding λ_{ij} values from above within their 'good' regions, where $y_* = \max_{c(r_i,a_j)} y_{ij}$, for a total of no more than $2|C_s| + (3 + y_*)|C_u|$ constraints. Solving MILPs is known to be $\mathcal{NP}$-complete, but it is difficult to formally characterize the hardness of solving this MILP due to the plethora of heuristics and pre-processing techniques utilized by commercial solvers like Gurobi (Gurobi Optimization, LLC, 2022), our solver of choice. Solve time depends not only on the numbers of (binary) variables and constraints in the formulation, but also on the tightness of the lower bound provided by solving its LP relaxation and the geometry of its feasible region, among other factors. Nevertheless, we have the following result:

Theorem 1. *Given a WPSTNU W, with (unknown) maximum expected value M, a solution $(b, \hat{B})$ to the MILP is found in $O(2^{|C_u|+|C_r|} LP(|T|))$ ($LP(n)$ is the complexity of solving a linear program over n variables) time such that $\hat{B} \leq M \leq \hat{B} + \hat{\epsilon}$.*

Thus, the exponential search described above is *sound* in that it will return a feasible schedule bounding below the maximum expected value, and *incomplete* insofar as it will return an optimal solution to the MILP given our piecewise linear approximation of the distributions of probabilistic durations of the WPSTNU, which nevertheless is a lower bound on the optimal schedule of the WPSTNU.

5.5 Limitations and Advantages of the MILP Formulation

We formulate the MILP in §5.1 without explicitly assuming that any pair of probabilities $P_i(\omega_i)$ and $P_j(\omega_j)$ in a WPSTNU are *mutually independent*. If the probabilistic durations are not independent, then the rewriting of the objective, and the approximation, depend on marginalizing out $P_i(\omega_i)$ and $P_j(\omega_j)$, and do not explicitly use the joint probability $P_{i,j}(\omega_i, \omega_j)$. The allocation of risk will generally increase the optimality gap. To see this, note that a schedule captures risk in an n-dimensional box, while correlated risk distributions could be covered by other 'shapes' with less restrictive constraints. In our evaluations, however, all $P_i(\omega_i)$ and $P_j(\omega_j)$ in a WPSTNU are mutually independent.

Allowing requirement constraints on two uncontrollables, $c(r_i, r_j)$, requires generalizing the functions $F_{ij}(x)$ to

$$\int_{x-u_{r_i,r_j}}^{x-l_{r_i,r_j}} P(\omega_j - \omega_i).$$

Even if $P_i(\omega_i)$ and $P_j(\omega_j)$ are unimodal, the composition, in general, is not (Ibragimov, 1956), and thus requirement constraints $c(r_i, r_j)$ are not permitted in our WPSTNUs with the exception of uncontrollables with normal distributions. The sum, and thus difference, of two independent normally distributed variables is a normal, and thus unimodal; under these and similar conditions the formulation provided works and constraints $c(r_i, r_j)$ are permitted. The same proviso applies when reformulating linked probabilistic duration constraints $d(a_i, r_i)$ and $d(r_i, r_j)$ [2]. We must replace $d(a_i, r_i)$ and $d(r_i, r_j)$ with $d(a_i, r_k)$ such that $P_k(\omega_k) = P_{i+j}(\omega_k)$. This is only possible when random durations are independent, and it's only easy to find P_{i+j} for certain kinds of distributions, e.g. normal but not beta distributions. As noted above, this means the MILP solution will underestimate the true expected value by a larger margin, increasing the error.

We also note that while our function bounding the probability of satisfaction from below does not define a concave region over its entire domain (and thus requires that a binary 'in-out' variable be added), it gives us the power to consider situations in which all possible outcomes $v(r_i)$ for an uncontrollable must fall entirely on one side of its mode. This was one of the difficulties in dealing with unimodal distributions that the authors in (Santana et al., 2016) did not address, as their choice of functions used to approximate risk required that the bounds for acceptable outcomes for an uncontrollable fall on either side of its mode. Our increased coverage comes at a price: the PARIS algorithm in (Santana et al., 2016) is polynomial-time, while solving the MILP is worst-case exponential, even when $C_r = \emptyset$ [3].

2. Allowed by HEATlab instances (Lund et al., 2017) but not (Tsamardinos, 2002) and thus not in ROVERS instances (Santana et al., 2016).

3. In practice, one probably would not want to satisfy a high-valued AR constraint with a probability of $< 50\%$, but the WPSTNU provides the opportunity to discover this feature of a schedule, and evaluate value changes to the problem instance that would further increase the probability of satisfaction.

6. Empirical Evaluation

We evaluate the correctness and computational effectiveness of our approach on the PSTNU instances of the ROVERS dataset generated by (Santana et al., 2016). Each instance of ROVERS features between 2 and 10 rovers performing between 1 and 10 planetary exploration tasks in parallel across a scheduling horizon before transmitting data within a shared communication window. Tasks to be completed by a single rover are generally modeled using contingent or probabilistic duration constraints; controllable timepoint a_i represents the start of task i and uncontrollable timepoint r_i represents its end. If task j follows task i, an intra-agent AR precedence constraint exists between r_i and a_j. Coordination between rovers is modeled by AR constraints $c(r_i, a_j)$, generally starting from uncontrollable timepoints, but also occasionally ending with them instead i.e. the 'non-canonical form' $c(a_i, r_j)$. These inter-agent constraints are only found within the communication window at the end of the schedule.

We also evaluate our approach using HEATlab instances generated by (Lund et al., 2017). These instances feature 20 timepoint variables divided among 2 to 4 agents and $20 - 35$ constraints depending on the selected interagent constraint density, of which up to 15 are contingent. Requirement constraints were set with no upper bound, and the total time (makespan) given to complete each schedule was the midpoint between the minimum and maximum possible times of the critical path through the PSTN. The interagent constraint density, i.e. the proportion of constraints between agents' sequences of actions, is variable. The degree of synchronization represents the "tightness" of the bounds on interagent constraints, which were set to $[0, n\sigma]$ with $n = 1, 2, 4$ and $\sigma \in [0, 5]$.

We modify the original benchmarks by adding different values $q_c(t_i, t_j)$ to the constraints, and stipulating some STN constraints as mandatory, as explained below. We also add additional constraints to make the instances more difficult, forcing more tradeoffs. We open this section with some analysis of the benchmarks, then describe our results.

Our piecewise linear approximation of distribution functions and MILP were implemented in Python, and the MILP was solved using Gurobi 8.1.1. The piecewise linear approx of F_{ij} used 50 pieces.

6.1 Checking for 'Strong Controllability'

The best expected value for an instance of EvSC will often depend on whether all of the STN constraints (rejectable and otherwise) can be satisfied. If this is not the case, then some constraint in C_r *must* be discarded; otherwise, whether to discard a constraint or not depends on the relative constraint values, as described in the introduction. As noted in (Fang et al., 2014), even such 'strongly controllable' instances may incur 100% risk. If an EvSC instance is 'controllable', an AR constraint may be implicitly rejected to maximize the expected value of other AR constraints, but we generally expect different performance of the MILP on these two classes of instances.

Note that the original ROVERS benchmark instances are PSTNUs for which all STN constraints must be satisfied. We characterize the MIT ROVERS benchmarks to determine if they are 'strongly controllable', by which we mean the STN constraints can all be satisfied with some minimum risk. Doing so allows us to compare the performance of our MILP based on the characteristics of the problem instances. Our check for strong controllability relies on the following lemma, adapted from the SC reduction for STNUs appearing in (Vidal & Fargier, 1999):

Lemma 2. *Given a PSTNU, a contingent link $g(a_i, r_i)$ with bounds $[l_{r_i}, u_{r_i}]$ and an AR constraint $c(r_i, a_j)$ with bounds $[l_{r_i,a_j}, u_{r_i,a_j}]$, a schedule s is guaranteed to satisfy $c(r_i, a_j)$ if and only if $s(a_j) - s(a_i) \in [u_{r_i} + l_{r_i,a_j}, l_{r_i} + u_{r_i,a_j}]$.*

Proof. We re-express $v(r_i)$ as $s(a_i)+\omega_i,\ \omega_i \in [l_{r_i}, u_{r_i}]$ so that in order to satisfy $c(r_i, a_j)$, we must have

$$\begin{aligned} & s(a_j) - (s(a_i) + \omega_i) \geq l_{r_i,a_j} \\ \iff & s(a_j) - s(a_i) \geq \omega_i + l_{r_i,a_j} \\ \iff & s(a_j) - s(a_i) \geq u_{r_i} + l_{r_i,a_j}. \end{aligned}$$

Similarly, we must have

$$\begin{aligned} & s(a_j) - (s(a_i) + \omega_i) \leq u_{r_i,a_j} \\ \iff & s(a_j) - s(a_i) \leq \omega_i + u_{r_i,a_j} \\ \iff & s(a_j) - s(a_i) \leq l_{r_i} + u_{r_i,a_j}. \end{aligned}$$

□

With this lemma in mind, we check for strong controllability of a PSTNU using the PSTN induced by removing all uncertain contingent links $g(a_i, r_i)$ and replacing their associated AR constraints $c(r_i, a_j)$ (or $c(a_j, r_i)$) with non-rejectable requirement constraints $c(a_i, a_j)$ whose bounds correspond to those of the lemma. If the resulting MILP admits one or more feasible solutions, the original PSTNU is strongly controllable; otherwise it is not, meaning at least one STN constraint must be discarded. Moreover, the schedule obtained by solving this MILP to optimality is the best strong schedule in terms of the value it expects to gain from satisfying probabilistic AR constraints, while ensuring all other constraints are satisfied. This may not be the optimal schedule when value is added and STN constraints can be rejected.

Using this check, we found a schedule for 3141 of the 4380 ROVERS instances. In 1067 of these optimal strong schedules, there was at least one AR constraint with $\lambda_{ij} = 0$, which implies no chance of satisfaction (i.e. the schedule had 100% risk). These numbers are higher than the number of controllable instances reported in (Santana et al., 2016) because our approach to truncating bounds on the probabilistic durations does not require them to 'straddle' the mode of the probability distributions of the probabilistic durations (as discussed in §5.1).

6.2 Computing 'Minimal Makespan'

Given a WPSTNU instance $<A,R,C,G,D,Q>$, we determined the minimum makespan required to achieve maximum expected value by first computing the maximum expected value $\hat{B}$ using the MILP presented in §5.1 and then solving the modified problem

$$\begin{aligned} & \min a_h \\ & \text{s.t. } (1) - (11) \\ & \quad \sum_{C_u} \lambda_{ij} \mathbf{q_c}(\mathbf{r_i}, \mathbf{a_j}) + \sum_{C_r} \beta_{ij} \mathbf{q_c}(\mathbf{a_i}, \mathbf{a_j}) \geq \hat{B}, \end{aligned}$$

where a_h is the variable representing the last scheduled timepoint of the WPSTNU (the horizon), which may be artificially added if one does not naturally exist.

Let the optimal makespan value we obtain from the MILP above be m^*. We will subsequently use fractions of m^* to squeeze schedule makespan by adding the constraint $a_h \leq \alpha m^*$ for some $\alpha < 1$ to the original MILP formulation. Adding these constraints will force additional tradeoffs in expected value in our modified benchmark problems. We can do this both for 'strongly controllable' problems as described above, and those where we already know some constraints must be rejected; we will not distinguish performance on the two types of instances when reducing the makespan.

6.3 Creating WPSTNU Benchmarks

In this section we present a series of empirical studies of our EvSC algorithm on WPSTNUs generated from the MIT ROVERS benchmark. We first replaced pairs of contingent links and AR constraints with a single STN constraint according to Lemma 2 (same as in the controllability check). We then classified all inter-agent STN constraints as rejectable and all others as non-rejectable, allowing us to find expected value-optimal schedules for the 'non-SC' instances described in §6.1. No lower bounds were imposed on the λ_{ij} values for the probabilistic AR constraints (meaning 100% risk, and zero expected value contribution, is a valid solution to these problems).

Because individual rovers (agents) within the ROVERS instances have sequences of alternating controllable and uncontrollable timepoints in the form $g(a_i, r_i)$,$c(r_i, a_j)$, the EvSC MILP can implicitly reject contingent constraint $g(a_i, r_i)$ by sacrificing the value on AR constraint $c(r_i, a_j)$ in order to schedule timepoint a_j *before* timepoint a_i, which violates the natural temporal ordering of a single rover's tasks. In order to prevent such illogical scheduling, we added an unrejectable, zero valued requirement constraint $c(a_i, a_j)$ with bounds $[0, \infty)$ between such sequences of controllable timepoints. This enforces the precedence between the intra-agent controllable timepoints, even when the AR constraints associated with contingent edges are implicitly sacrificed by the scheduler.

We created new derived benchmarks from the ROVERS instances in stages. First, we describe changes to makespan and the number and type of constraints added to the original ROVERS instances. Next, we describe different *value schemes*, corresponding to how we add values q_c to the constraints in the ROVERS benchmarks. The combination of structural modifications and value schemes is used to create many different derived benchmarks.

Our first modification is to reduce the makespan by up to 50% compared to the minimum, as described above. In our second modification, we prioritized coordination, so inter-agent constraints are valued more highly than intra-agent constraints. In order to drive increased search costs by introducing more choices, we modified the original benchmark by adding low-value inter-agent constraints that must be traded off against the high-value intra-agent constraints. Allowing inter-agent constraints to be removed lets us explore trade-offs between individual rovers completing their assigned tasks, and forcing pairs or groups of rovers to cooperate with one another. The new constraints were realized as rejectable STN constraints of the form $c(a_i, a_j) = [-5, 5]$. Our decision to use this comparatively small temporal range was due to the arbitrariness of some of these random constraints with respect to the originally constructed instances. For our third modification, in order to explore the consequences of tight makespan constraints, we reversed the prioritization, making the intra-agent AR constraints more valuable than inter-agent constraints. To explore the impact of skew probability distributions, we modified the original benchmark to make probabilistic AR constraints more valuable than uncertain AR constraints; STN constraints remain low value. To implement the

Value scheme	Value of constraints (q_c)		
	IRA_o	IRA_n	ITA
1	5	5	1
2	1	-	3
3	5	1	1
Value scheme	Value of constraints (q_c)		
	pAR	uAR	STN
4	3	1	1

Figure 7: Value schemes used for all experiments. For the first three value schemes, IRA_o refers to inter-agent constraints in the original instance, IRA_n refers to new, randomly added inter-agent constraints, ITA refers to intra-agent AR constraints. For value scheme 4, pAR refers to probabilistic AR constraints, and uAR refers to uncertain AR constraints.

Derived Benchmark	Distribution	Added Constraints	Makespan Constraint	Value Scheme
1	normal	0	none	1
2	normal	0	$50\% - 100\%$	2
3	normal	$0 - 50$	100%	3
4	beta	0	100%	4

Figure 8: Parameters of the ROVERS derived benchmarks used to evaluate the Strong Controllability MILP.

removability of AR constraints, we ensured the STN constraints created using Lemma 2 had the correct value and were labeled rejectable. This scheme differs from the previous benchmarks in that the 'agent' structure of the problem is not used to drive constraint value.

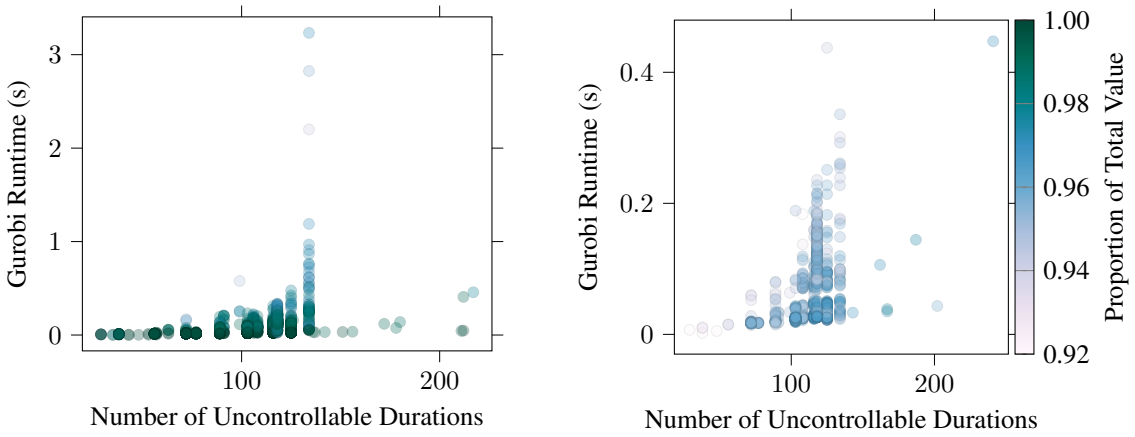
Each derived benchmark is accompanied by a value scheme to augment the ROVERS instances with constraint qualities $q_c(t_i, t_j)$. The four different constraint value schemes are described below:

1. All inter-agent constraints, both STN and AR, are more valuable ($q_c = 5$) than their intra-agent counterparts ($q_c = 1$).

2. Inter-agent constraints, both STN and AR, have a mix of values ($q_c \in \{1, 5\}$); constraints in the original problem (IRA_o) have value 5, constraints not in the original problem (IRA_n) have value 1. Intra-agent constraints are valued at 1.

3. Intra-agent AR constraints (ITA) are more valuable ($q_c = 3$) than inter-agent constraints ($q_c = 1$).

4. Probabilistic AR constraints (pAR) are more valuable ($q_c = 3$) than uncertain AR (uAR) constraints ($q_c = 1$). Since we replace uncertain links with STN constraints, those replaced STN constraints are rejectable and have $q_c = 1$. All other inter-agent STN constraints were given $q_c = 1$.



(a) Solve times and proportions of total value expected to be obtained for strongly controllable instances on derived benchmark 1.

(b) Solve times and proportions of total value expected to be obtained for non-strongly controllable instances on derived benchmark 1.

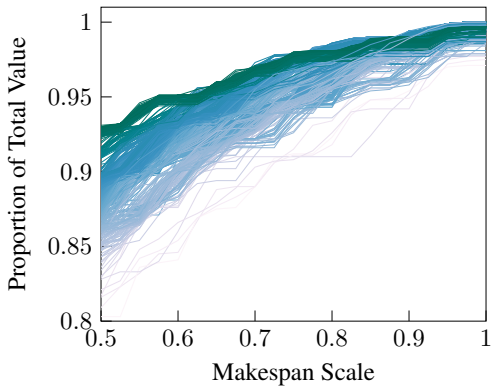
Figure 9: Solve times and proportion of total expected value for both controllable and uncontrollable instances from ROVERS derived benchmark 1. The color gradient applies to both figures.

The value schemes are summarized in Figure 7. Our derived benchmarks for the ROVERS instance each use one of the four value schemes, in conjunction with either reduction in makespan, addition of new constraints, or exchanging normal distributions for beta distributions. The derived benchmarks are shown in Figure 8.

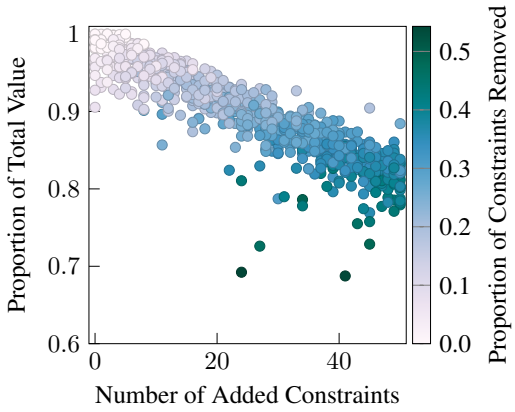
6.4 Evaluating the MILP: ROVERS

We first report results on derived benchmark 1. All benchmarks are run on a Linux laptop with an Intel 4-core i7-8550U CPU with 16 Gb of RAM. Figures 9a and 9b plot runtime versus the number of uncontrollable durations; the color map indicates the proportion of total value achieved. Figures 9a shows results for the 'strongly controllable' instances of the ROVERS benchmark, and Figure 9b shows results for the 'uncontrollable' instances, where 'controllable' and 'non-controllable' instances are checked as described in §6.1. The EvSC MILP produced a schedule expected to obtain nearly all ($> 95\%$) of the possible value from AR constraints for most of the ROVERS instances, even those that aren't strongly controllable, with solve times under 0.5 seconds. Note the solve times y-axis ranges in Figures 9a and 9b differ due to a few outliers in Figures 9a. Solve times are comparable to those reported by (Santana et al., 2016) when solving the easier risk minimization version of the original ROVERS benchmark.

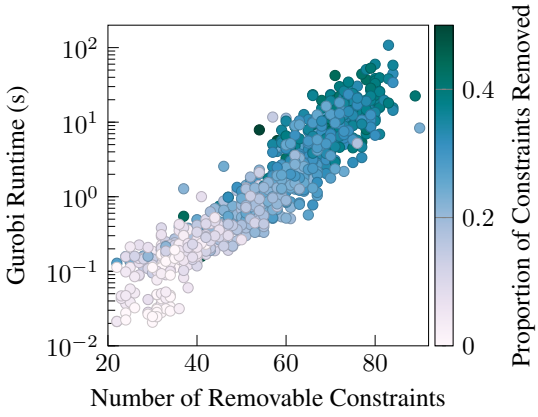
In order to assess the quality of our MILP solution, we calculate the error as described in §5.3. We report the quality of solution $\frac{\hat{B}}{B}$, where B is the true expected value ($B = \hat{B} + \epsilon(b)$) of the optimal schedule returned by the MILP, as a proxy for the error relative to the true quality $\frac{\hat{B}}{M}$. For derived benchmark 1, the mean value of $\frac{\hat{B}}{B}$ was 0.999999668 and the minimum (maximum error) was 0.999992589. Unsurprisingly, optimal schedules for strongly controllable instances obtained a slightly higher proportion of their total value than non-strongly controllable ones.



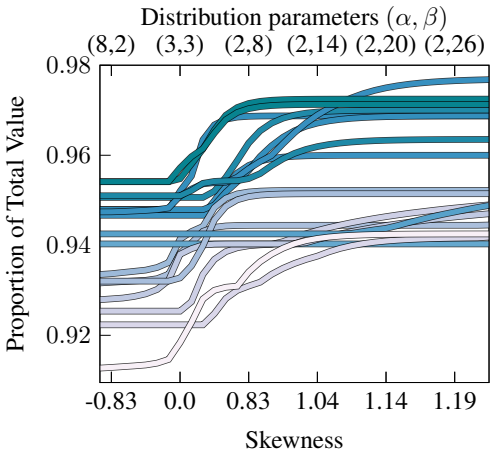
(a) Proportions of total value when limiting makespan over 876 (1 in every 5) instances on derived benchmark 2 as makespan decreased from 100% to 50% of minimum.



(b) Proportions of total value and proportions of constraints removed as a function of the number of randomly added concurrency constraints on derived benchmark 3.



(c) Solve times and proportions of constraints removed as a function of the total number of rejectable constraints (original plus added) on derived benchmark 3.



(d) Proportions of total value as a function of contingent edge distribution skewness over a randomly selected subset of derived benchmark 4.

Figure 10: Performance of the MILP solver on ROVERS derived benchmarks 2,3 and 4.

In order to further explore the aforementioned trade-offs, we now discuss results on the remaining derived benchmarks. Derived benchmark 2 forces more tradeoffs between rejectable inter-agent constraints and probabilities of AR constraint satisfaction by artificially limiting schedule time to completion (makespan) while encouraging the preservation of intra-agent constraints. Figure 10a plots the impact of incrementally cutting allowed makespan from 100% down to 50% of its minimal value, as computed using the MILP described in §6.2, over a representative subset (one in every five) of the instances. It is interesting to note that even severely limiting schedule makespan to half of its minimal value did not have a major effect on optimal expected value for many of the tested instances; often, 90% of the expected value can still be achieved. Runtime on these problems remained low, mostly under one second.

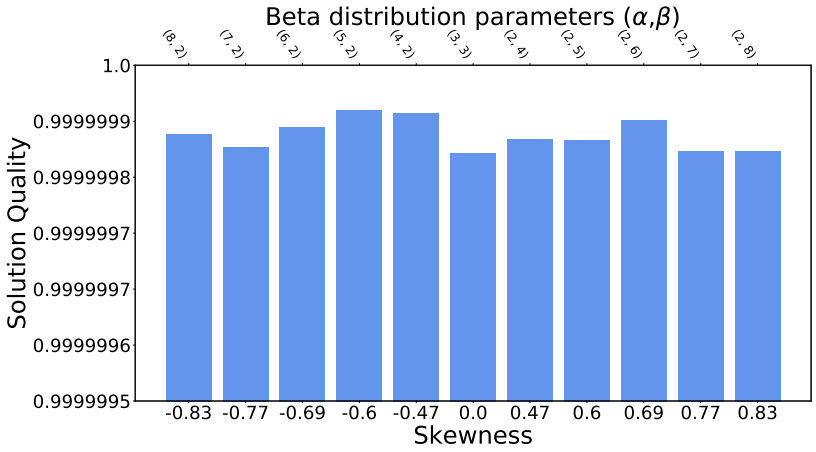


Figure 11: Solution quality as a function of skewness averaged over derived benchmark 4.

Derived benchmark 3 randomly introduces additional inter-agent concurrency constraints to simulate higher levels of coordination between rovers. The largest ROVERS problem might have 10 rovers and on average 5 activities each with 10 coordination constraints at the end for about 60 constraints; we add between 0 and 50 extra rejectable constraints. Figure 10b shows how the number of added concurrency constraints impacted expected value for the same subset of instances examined in Figure 10a. The color gradient shows the proportion of rejectable requirement constraints C_r (both added and original) rejected in the optimal schedule, presumably in the interest of increasing the probabilities of satisfying AR constraints (due to their low value). Adding these constraints reduces the percentage of total value achieved, which is not surprising; at 50 added constraints, the proportion of total value achieved drops below 80%. Adding these constraints also significantly increased solve times; Figure 10c plots solve time as a function of the number of rejectable constraints for the same subset of instances, showing that the EvSC MILP becomes much more difficult to solve as more rejectable constraints encouraging coordination are included. Still, all problems are solved in under two minutes.

In derived benchmark 4, we experimented with converting probabilistic edges with normal (μ, σ) distribution into beta distribution with lower and upper bounds of $x - 2\sigma$ and $x + 2\sigma$, respectively, where the center x is chosen so that the beta distribution has mean μ. The parameters α and β are chosen to skew all distributions to the left or right. Because the beta distribution functions we chose are unimodal with a large convex region, our linear lower bounding strategy was still applicable. Figure 10d plots change in expected value when varying α and β parameters on derived benchmark 4. Recall this benchmark values probabilistic AR constraints over all other constraints. As evidenced by the figure, the proportion of total value is larger for problems in which the beta distributions skewed right (low durations are more likely). Because the ROVERS instances generally have more lower-bound constraints than upper-bound constraints, it is unsurprising that as probabilistic durations tend toward shorter outcomes, the optimal schedule is able to account for more probability mass and therefore obtain greater expected value. The typical loss of percentage of total value for the randomly selected problems due to skew ranges from $2\% - 3\%$ (skew right) to $6\% - 8\%$ (skew left).

Derived Benchmark	Distribution	Added Constraints	Makespan Constraint	Value Scheme
5	normal	0	none	1
6	normal	0	$50\% - 100\%$	2
7	normal	$0 - 30$	100%	1
8	normal	$0 - 30$	100%	3

Figure 12: Parameters of the HEATLab derived benchmarks used to evaluate the Strong Controllability MILP.

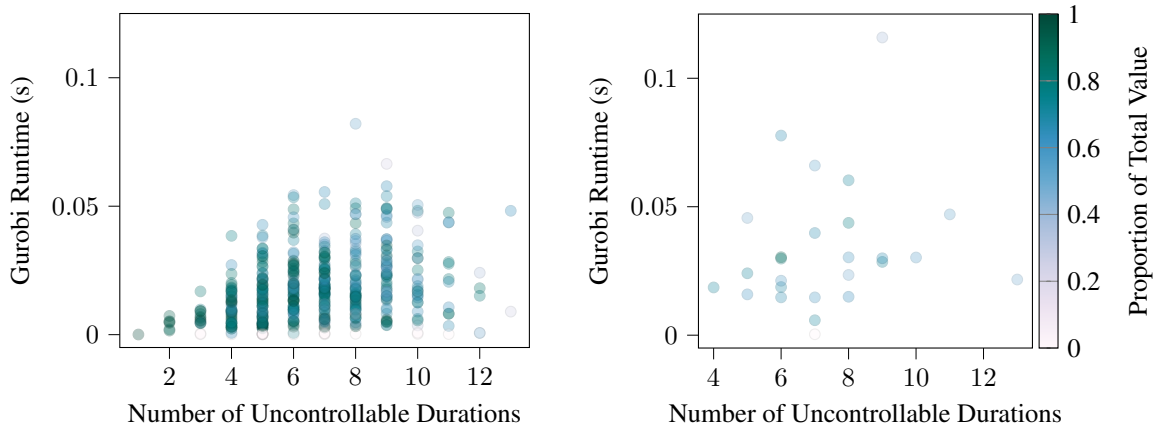
We again used derived benchmark 4 to evaluate the impact of skewness on solution quality. As we did with normal distributions, we evaluate $\frac{\hat{B}}{B}$ with B being the true expected value of b over problem instances with skew distributions. Over the skew distribution instances, the error incurred in the objective value because of λ underestimates gives a solution quality with a mean $\frac{\hat{B}}{B}$ of 0.9999999 and a minimum of 0.99999984. Figure 11 shows the average error on derived benchmark 4 as a function of skewness. The addition of skewness does not impact the error of our solutions. While the piecewise linear bounds will certainly truncate the tails of F_{ij}, these results suggest that the probabilities found for the ROVERS benchmarks are well inside the concave region of F_{ij} where we expect to see low error due to our approximation.

6.5 Evaluating the MILP: HEATlab

We perform an evaluation on the HEATlab dataset (Lund et al., 2017) analogous to the evaluation discussed in §6.1. Derived benchmarks for the HEATLab are shown in Figure 12. Derived benchmark 5 is the original HEATLab dataset with value scheme 1. Derived benchmark 6 crushes the makespan, but uses value scheme 2. Derived benchmarks 7 and 8 use different value schemes with added intra-agent constraints, for reasons explained below. The HEATlab instances have chained uncontrollable durations, making it difficult to evaluate the implications of beta distributions on the results, since beta distributions are not closed under summation; therefore we do not experiment with beta distributions for the HEATlab dataset.

Figures 13a and 13b plot runtime versus the number of uncontrollable durations for 'strongly controllable' and 'uncontrollable' instances of derived benchmark 5, respectively. The controllability of these instances was checked as described in §3. Since HEATlab instances tend to be much smaller than ROVERS instances, the runtime for computing optimal solutions for HEATlab instances tends to much lower. The runtime for computing optimal solutions using our algorithm is about an order of magnitude faster than the SREA algorithm (Lund et al., 2017). As demonstrated by the difference in color gradient scale between Figure 9 and 13, optimal solutions for the HEATlab instances achieve significantly lower values than ROVERS instances. Over derived benchmark 5, error incurred in the objective value because of λ underestimates gives a solution quality with a mean $\frac{\hat{B}}{B}$ of 0.9997 and a minimum (maximum error) of 0.9925.

Figure 14a plots the impact of incrementally cutting the allowed makespan from 100% down to 50% of its minimal value, as computed using the MILP described in §6.2. Crushing makespan over the HEATlab dataset seems to have a similar effect on both HEATLab and ROVERS datasets. One major difference between the two datasets seems to come from the fact that HEATlab instances



(a) Solve times and proportions of total value expected to be obtained for strongly controllable instances on derived benchmark 5.

(b) Solve times and proportions of total value expected to be obtained for non-strongly controllable instances on derived benchmark 5.

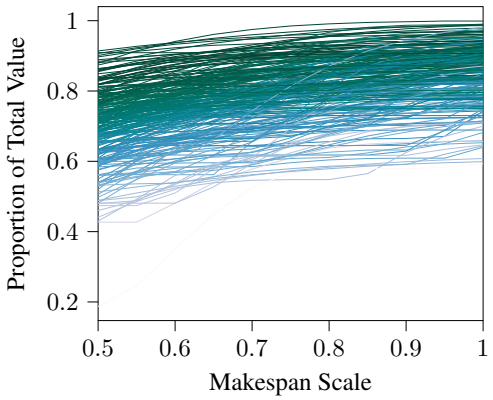
Figure 13: Solve times and proportion of total expected value for both controllable and uncontrollable instances from HEATlab derived benchmark 5. The color gradient applies to both figures.

achieve a significantly lower proportion of total value than ROVERS instances. Even accounting for this, it appears crushing the makespan significantly reduces the achievable value when compared to the ROVERS benchmark. Crushing the makespan reduces the % of total value to $40 - 90\%$ of the max value when the makespan is reduced by 50% at worst, when compared to ROVERS, where the worst case is 80%.

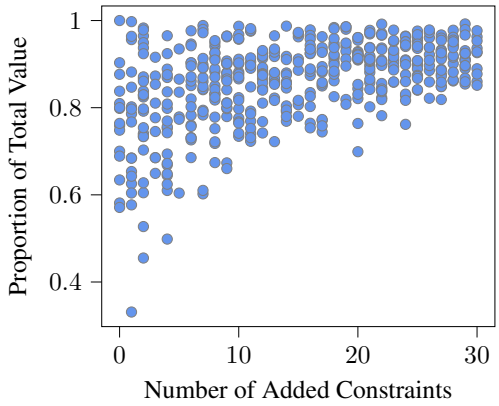
Similar to derived benchmark 3, derived benchmark 7 introduces additional inter-agent concurrency constraints using value scheme 1. Figure 14b shows the results. We remark that added constraint bounds $[-5, 5]$ are actually loose constraints for HEATlab, as opposed to tight constraints for ROVERS, since the makespan of HEATlab instances tends to be quite small (between $10 - 30$). We see that the proportion of total value is quite variable, with the worst case again lower than we found for ROVERS instances. We note there is only a weak dependence of the worst-case proportion of total value achieved on the number of constraints added. Curiously, it appears that *more* total value is achieved in the worst case as more constraints are added; this may be because the constraints are comparatively loose, and thus are more likely to be satisfied. Finally, we note there is little variation in the number of rejected constraints regardless of the number of constraints added to HEATlab problems, which differs from the behavior of ROVERS problems with added constraints. Figure 14c shows runtime variation in the presence of added constraints on derived benchmark 7. Here, we see that most runtimes are below 0.2 seconds, but that the worst-case runtime grows as more constraints are added. Again, the runtime impact of adding more constraints is weaker than we saw for ROVERS benchmarks with added constraints.

We also generated derived benchmark 8, a slightly modified version of derived benchmark 7 that uses value scheme 3. This value scheme reduces the value of added interagent constraints from 5 to 1 and the resulting plot in Figure 14d shows minimal correlation between the number of added constraints and the proportion of total value. This result corroborates the hypothesis that the extra proportion of total value achieved in Figure 14b likely comes from the relative looseness of the

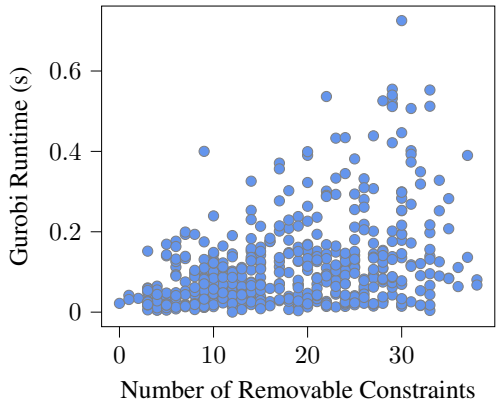
added constraints on the HEATLab benchmark, making them easy to satisfy. Runtime for derived benchmark 8 was comparable to that for derived benchmark 7 and thus is not shown.



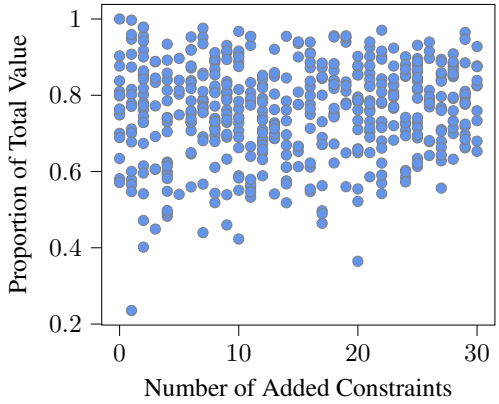
(a) Proportions of total value when limiting makespan over a randomly chosen half (for legibility) of the instances on derived benchmark 6 as makespan decreased from 100% to 50% of minimum.



(b) Proportions of total value as a function of the number of randomly added concurrency constraints on derived benchmark 7.



(c) Solve times as a function of the total number of rejectable constraints (original plus added) on derived benchmark 7.



(d) Proportions of total value as a function of the number of randomly added concurrency constraints on derived benchmark 8.

Figure 14: Performance of the MILP solver on HEATlab derived benchmarks 6,7, and 8. The heatmaps are hidden from plots 14b and 14c since the proportion of constraints removed from these benchmarks tended to be very low with a few outliers that did not show any meaningful relationships. An analogous plot for 10d, proportion of total value as a function of skewness, is not provided because the HEATlab instances contain sequences of probabilistic edges, which are difficult to evaluate since beta distributions are not closed under summation.

7. Rescheduling Approaches

We have so far focused our attention on generating fixed schedules, the analog of strong controllability (SC) in STNUs. Simply executing such a schedule does not take advantage of information gained during execution, which could improve the expected value. In order to motivate the benefits

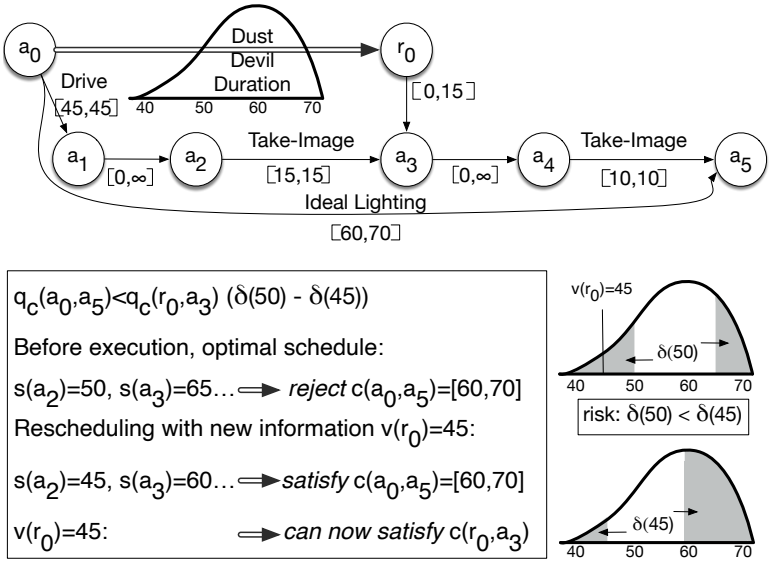


Figure 15: Rescheduling in presence of new information during execution can improve schedule expected value. Before execution, the ideal lighting constraint for the second image was rejected on the expectation that the higher (expected) value dust devil would be observed. When new information becomes available, changing the schedule can satisfy both the ideal lighting constraint *and* observe the dust devil.

of rescheduling during execution in the setting of WPSTNUs, we revisit the dust-devil example from Figure 1. Figure 15 illustrates rescheduling in the presence of new information. As before, let schedule s' be the optimal fixed schedule with $s'(a_2) = 50$ that violates a single constraint, namely, $c(a_0, a_5)$, in order to maximize the probability, and thus the expected value, of $c(r_0, a_3)$. Unlikely though it may be, if the dust devil ends at just the right time, i.e. $v(r_0) = 45$, a schedule can satisfy *both* $c(r_0, a_3)$ and $c(a_4, a_5)$. Even though the mandatory drive time is 45, followed by the mandatory take-image time of 15, and since $c(r_0, a_3) \in [0, 15]$, it is possible to take the dust-devil image by scheduling $s(a_2)= 45$, and $s(a_3)= 60$, barely satisfying $c(r_0, a_3)$. Further, it is possible to schedule $s(a_5)= 70$, and therefore satisfy the ideal lighting constraint $c(a_0, a_5)$ on the second image acquisition task.

By contrast, suppose $v(r_0) = 44$. Because the dust devil now ends one time unit before the drive ends, and because $c(r_0, a_3) \in [0, 15]$, it is no longer possible to satisfy $c(r_0, a_3)$; $a_3 \geq 60$ for all schedules since the drive is mandatory, and $60 - 44 = 16$. However, it is still possible to schedule $s(a_5)= 70$, and therefore satisfy $c(a_0, a_5)$, which still requires scheduling $s(a_2)= 45$. Sticking to the original schedule s' would satisfy neither constraint $c(r_0, a_3)$ nor $c(a_0, a_5)$.

More intriguing rescheduling opportunities include rescheduling after the passage of time but without observing events, and rescaling the probability distributions of unobserved uncontrollable timepoints, to recompute expected value tradeoffs.

While there is certainly an analog of dynamic controllability (DC) for WPSTNUs i.e. the Expected Value Dynamic Controllability (EvDC) problem, it is not clear how best to pose and solve such a problem. In lieu of achieving this goal, we take inspiration from (Lund et al., 2017) and (Abrahams et al., 2019), who employ *rescheduling* to improve solvability rates of PSTNs. Reschedul-

ing involves execution of a schedule s, observing uncontrollable event outcomes, and re-posing and re-solving PSTNs to satisfy constraints, or reduce or redistribute risk. During execution of an STNU, a controllable timepoint a_i is called *enabled* at time t if executing it at t does not violate any constraints; that is, t is later than the earliest start time of a_i and all a_j that must precede a_i have executed. An uncontrollable timepoint r_i from a duration constraint $d(a_i, r_i)$ or contingent constraint $g(a_i, r_i)$ is called *enabled* after controllable a_i is executed. When PSTNs are transformed into STNUs, enablement is used as a trigger for rescheduling. DREA (Lund et al., 2017) reschedules after any uncontrollable timepoints become enabled, or are observed, and accepts the new schedule unconditionally. By contrast, the approaches described in (Abrahams et al., 2019) reschedule and accept the newly generated schedule under conditions designed to reduce DREA's schedule invocations and communications, and hence resource usage, at the expense of reduced success rate for PSTNs.

Rescheduling from scratch can be time and resource intensive. An alternative approach to responding to new information is to use a low-order computation time algorithm to select the next timepoint to execute, rather than reschedule. (Saint-Guillain, Vaquero, Agrawal, & Chien, 2020) propose generalizing such approaches to selecting the next timepoint to execute, referring to them as Dispatch Protocols (DPs), and describe methods to measure the risk for PSTNs of low computation time DPs that occupy a mid-point between SC and DC.

In this paper, we explore a class of *rescheduling approaches* during execution to improve expected value for WPSTNUs. The interleaving of execution and rescheduling generally proceeds as follows:

1. Execution begins with a schedule s.
2. Over time, controllable timepoints are executed according to their scheduled times $s(a_i)$, and uncontrollable event outcomes $v(r_i)$ may be observed.
3. The information gleaned during the execution of the schedule is used to determine whether or not to reschedule. If rescheduling occurs, a new schedule s' is generated.
4. The current schedule s and new schedule s' are evaluated to determine whether or not to continue executing s or switch to the new schedule s'.

In this paper, we limit our investigation to *fixed-period* rescheduling approaches. Fixed-period approaches are simple to implement, but we show that they nevertheless can led to significant improvement in expected value when compared to executing the fixed SC schedule. We distinguish *time-based* approaches, which reschedule after a fixed amount of time has passed, regardless of the number or type of timepoints that are executed or observed, and *event-based* approaches, which reschedule after a fixed number of timepoint events are observed.

Our interest is in understanding the improvement in expected value resulting from rescheduling methods and frequency. Each method has pros and cons, as we now show. Event-based rescheduling permits less frequent re-normaliziation of the probability of uncertain durations than time-based rescheduling. Notably, enabled uncontrollables (those whose controllable timepoint has been executed) will only be re-normalized when executing a scheduled controllable, or when another enabled uncontrollable ends. Time-based approaches allow for re-normalizing probabilities of enabled uncontrollable events at regular intervals, increasing responsiveness to changes in the expected value of AR constraints. However, fixed-period time-based rescheduling cannot instantaneously react when

controllables are enabled or uncontrollables occur. If the period between rescheduling is too long, then opportunities to reschedule will be lost, resulting in lower expected value. Short rescheduling periods mitigate this, but are inefficient. Event-based approaches have a rough bound on the number of times new schedules can be generated, namely, $|T| = |\mathrm{A}| + |\mathrm{R}|$. This bound is imprecise because we allow a controllable that is enabled, and about to execute, to be *postponed* by rescheduling. This means each controllable could be enabled multiple times prior to execution. Time-based rescheduling approaches, by contrast, permit an amount of rescheduling proportional to the scheduling horizon, which will generally exceed the number of timepoints. This mix of advantages and liabilities of the approaches makes it unclear which class of approach is superior.

Previous approaches to rescheduling either unconditionally accept the newly generated schedule (Lund et al., 2017), or accept a new schedule for PSTNUs only if its risk is sufficiently reduced compared to the risk of the current schedule (Abrahams et al., 2019). As we will see, the MILP approximation introduced in §6 also leads to the need to evaluate a new schedule for WPSTNUs prior to accepting it, because it might not be *provably better* than the current schedule.

7.1 Definitions

We formalize the elements of rescheduling approaches for WPSTNUs as follows. We first define *fixed-period rescheduling policies*, whose role is to decide whether to reschedule or not. Time-based fixed-period rescheduling policies are simple to characterize: we reschedule every k time units, regardless of what events occur. Fixed-period event-based policies require more careful definition of the events defining the rescheduling period. We define the events over which fixed-period rescheduling is characterized. Once rescheduling policies are described, we define *selection criteria*, the conditions under which we switch from the current schedule to a newly generated one. With this formal foundation, we can then define rescheduling approaches as a combination of a scheduler, rescheduling policy, and a selection criterion.

We begin with fixed-period time-based rescheduling policies:

Definition 10. *Given a WPSTNU W and a schedule s. We refer to the set of possible unit clock ticks by Γ (essentially, $\Gamma \equiv \mathbb{Z}^+$). A* fixed-period time-based rescheduling policy $\pi_{\Gamma,k}$ *reschedules every k unit clock ticks.*

Note that time-based policies reschedule regardless of whether an event has occurred or is slated to occur at t. We can rescale time to admit arbitrarily small rescheduling periods k and avoid the difficulties of real-valued time.

We now define event-based rescheduling policies. To do so, we generalize enablement of controllable timepoints to permit violation of constraints, which means we can use a simpler definition than that of (Lund et al., 2017); enablement of uncontrollables remains unchanged. Enablement of an uncontrollable timepoint r_i only happens once, when a_i is executed. As we will see below, enablement of the same controllable timepoint a_i may happen multiple times during execution if a_i is postponed by rescheduling.

Definition 11. *Given a WPSTNU W and a schedule s. Let t be the current time during the execution of the schedule. Controllable timepoint a_i is* enabled *at time t if $t = s(a_i)$. Uncontrollable timepoint r_i in contingent constraint $g(a_i, r_i)$ or duration constraint $d(a_i, r_i)$ is* enabled *if a_i was executed at time $u \leq t$ and has not been observed.*

Definition 12. *Given a WPSTNU W. Define the set of all* timepoint events *E that can occur during execution and rescheduling of W as $E = \{n(a_i) : a_i \in A\} \cup \{v(r_j) : r_j \in R\}$, where $n(a_i)$ denotes enablement of controllable timepoint a_i and as before, $v(r_j)$ denotes observation of uncontrollable timepoint r_j. Let X denote an* event sequence *of events in E, and denote the set of all event sequences X over E by $\mathcal{E}$. An* event-based rescheduling policy *is a Boolean function $\pi : \mathcal{E} \to \{\top, \bot\}$.*

Referring again to the Dust Devil example in Figure 15, the set of timepoint events is $E = \{n(a_i), i \in 1 \dots 5 \cup v(r_1)\}$. Recall optimal schedule s' for the initial Dust Devil problem before execution assigns $s(a_1)= 0$ and $s(a_1)= 50$. If r_1 is observed at $t = 51$, the sequence ending when r_1 is observed is $X = \{n(a_1), n(a_2), v(r_1)\}$.

An event-based rescheduling policy indicates whether rescheduling should occur after having observed an event sequence during the execution of the schedule. The time t of the last event in the sequence isn't used in defining the rescheduling policies. Note that a sequence is not just a perturbation of a (subset of) events in E, because controllable timepoints can be enabled multiple times if they are postponed by rescheduling. We don't define event-based rescheduling policies based on enablement of uncontrollable timepoints, because Definition 11 ensures that the execution of controllables enables uncontrollables. Finally, we don't include the execution of timepoints in the set of events, because controllable timepoints are executed immediately when they are enabled, unless the rescheduler moves them.

We now define two *fixed-period event-based rescheduling policies* $\pi_{T,k}$ and $\pi_{R,k}$:

Definition 13. *Given a WPSTNU W and a schedule s. Let E be the set of events for W. Let X be a sequence of events as defined above, and let $N(X)$ and $V(X)$ denote the numbers of controllable timepoint enablements and uncontrollable timepoint observations, respectively, included in X.*

Define the fixed-period event-based rescheduling policy $\pi_{T,k}(X)$ *as follows:*

$$\begin{cases} \pi_{T,k}(X) = \top & \textit{if } (N(X) + V(X)) \bmod k = 0, \\ \pi_{T,k}(X) = \bot & otherwise. \end{cases}$$

Rescheduling occurs if $\pi_{T,k}(X) = \top$. The subscript T emphasizes that rescheduling can occur at an event involving any timepoint in $T = A \cup R$.

Let e be the last event in X. Define the fixed-period event-based rescheduling policy $\pi_{R,k}(X)$ *as follows:*

$$\begin{cases} \pi_{R,k}(X) = \top & \textit{if } (V(X) \bmod k = 0) \wedge (\exists r_i | e = v(r_i)), \\ \pi_{R,k}(X) = \bot & otherwise. \end{cases}$$

Under $\pi_{R,k}$, we reschedule if $\pi_{R,k}(X) = \top$. The subscript R emphasizes that rescheduling occurs only at events involving timepoints in R.

The intuition behind fixed-period event-based policies is that rescheduling occurs every k events. Unlike fixed-period time-based policies, rescheduling only occurs in response to events. Policy $\pi_{R,k}$ includes the condition that the last event e in the sequence is an observation of an uncontrollable to ensure rescheduling doesn't occur every time a controllable is executed after $(V(X) \bmod k = 0)$. Referring again to the Dust Devil example in Figure 15, consider sequence $X = \{n(a_1), n(a_2), v(r_1)\}$.

Under policy $\pi_{R,1}$, rescheduling would occur after this sequence is observed because $V(X) = 1$, $1 \mod 1 = 0$, and the last event is an observation of an uncontrollable timepoint. Under policy $\pi_{T,3}$ rescheduling would also occur after this sequence is observed because $N(X) + V(X) = 3$, and $3 \mod 3 = 0$. Under policy $\pi_{T,1}$, rescheduling would occur after this sequence is observed because $N(X) + V(X) = 3$ and $3 \mod 1 = 0$. Finally, under policy $\pi_{T,2}$, rescheduling would not occur after this sequence is observed, because $N(X) + V(X) = 3$, and $3 \mod 2 = 1$; however, rescheduling would occur under $\pi_{T,2}$ after observing sequence $X = \{n(a_1), n(a_2)\}$.

As mentioned previously, rescheduling at time t is performed *before* any events for which $s(a_i) = t$ are executed for all rescheduling policies we consider, and could postpone events for which $s(a_i) = t$. This means event-based rescheduling could occur more than $\frac{|T|}{k}$ times. Also note rescheduling is assumed to be 'instantaneous', meaning that even if $s(a_i) = u > t$, the current event can be rescheduled to the current time, i.e. $s(a_i) = t$, and immediately executed. Finally, suppose the event count is $j < k$, but more than $k - j$ events occur simultaneously. Our approach is to reschedule when the event count reaches k; events occurring at exactly the same time are selected arbitrarily to reach k events prior to rescheduling. In large problems and 'dense' time real systems, especially with many uncontrollable events, this corner case is unlikely to occur. Due to the non-determinism of uncontrollable event occurrences, we believe a more elaborate deterministic strategy is unlikely to lead to substantively different results than those we report below.

For this paper, the scheduling algorithm is the MILP previously presented in §6, which will be used both to generate the initial schedule, and to reschedule during execution; we denote this MILP approach by M.

In the next section we describe the design of the selection criteria, but provide our notation here:

Definition 14. *Given a WPSTNU W, a current schedule s, and a proposed schedule s'. A* selection criterion $\chi : S \times S \to S$*, where S is the set of all possible schedules, chooses one of the two schedules s, s' to continue executing.*

Fixed-period rescheduling approaches are thus parameterized by a scheduling algorithm M, a fixed-period rescheduling policy $\pi_{Y,k}$ with $Y \in \{\Gamma, T, R\}$, and a selection criterion χ, and are denoted $\mathcal{R} = (M, \pi_{Y,k}, \chi)$. After an initial schedule s is generated by M, execution of the schedule commences. While executing s, the policy $\pi_{\Gamma,k}(X)$ tracks time; the policies $\pi_{T,k}(X)$ and $\pi_{R,k}(X)$ count subsets of events. If enough time has passed, or enough events have occurred, then rescheduling using M is invoked, and a new schedule s' is generated based on the current time, executed controllables, and uncontrollable occurrences. Once the new schedule is generated, the selection criterion $\chi(s, s')$ determines whether to continue executing s, or replace s with s', after which execution resumes.

Now that we have defined rescheduling approaches, we turn to the expected value proposition. From Definition 9, we know how to evaluate the value of a fixed schedule s and a particular outcome ω:

$$f(s, \omega) = \sum_{c \in C} q_c(t_i, t_j)\ \sigma(c(t_i, t_j), s, \omega).$$

This definition can be extended to capture the expected value of a rescheduling approach $\mathcal{R}$ for a particular outcome, $f(\mathcal{R}, \omega)$. Rescheduling approach $\mathcal{R}$ will, ultimately, produce some as-executed schedule s_e, which depends on ω, but otherwise, the value function is well-defined, i.e.,

$$f(\mathcal{R}, \omega) = \sum_{c \in C} q_c(t_i, t_j)\ \sigma(c(t_i, t_j), s_e, \omega).$$

This, in turn, means that the expected value $g(\mathcal{R})$ is also well-defined, leading to the following:

Definition 15. *Let W be a WPSTNU. A* rescheduling approach *for W is a 3-tuple $\mathcal{R} = (M, \pi, \chi)$. Denote the value obtained by $\mathcal{R}$ given a particular vector $\omega \in \Omega$ of outcomes of uncontrollable timepoints by $f(\mathcal{R}, \omega)$. Then the value obtained by $\mathcal{R}$ has expectation $g(\mathcal{R}) = \int_{\omega \in \Omega} f(\mathcal{R}, \omega) P(\omega)$.*

We now compare the Dynamic Robust Execution Algorithm (DREA) of (Lund et al., 2017) to our rescheduling approaches. We observe that DREA's policy is reminiscent of rescheduling policy $\pi_{T,1}$ applied to PSTNUs. DREA reschedules when uncontrollable events are first enabled or are observed, or when a controllable executes while another uncontrollable is enabled but hasn't yet been observed. DREA does not reschedule if a controllable is executed but no uncontrollables are enabled. By contrast, $\pi_{T,1}$ reschedules when any controllable is enabled or uncontrollable timepoint is observed. DREA reschedules only after executing controllable timepoints, without the option to 'postpone' them. DREA's scheduler is an LP, and the selection criterion is simply to accept the new schedule, while our rescheduling approaches use a selection criterion to choose whether to keep a schedule or not.

7.2 Designing a Selection Criterion

In this section we explain how to design a selection criterion χ that is *guaranteed* to improve the expected value. To set the stage for this selection criterion, we start with a simple example.

Let s be an arbitrary schedule generated for WPSTNU W prior to execution. Consider a rescheduling approach $\mathcal{R} = (M, \pi_t, \chi)$ where scheduling policy π_t employs a scheduler M to suggest an updated schedule s_t exactly once at an arbitrary time t, and a selection criterion χ that decides whether to switch the schedule from s to s_t. We observe that the sequence X incorporates all timepoint executions and observations that occur before t; implicit in X are constraints on executed controllables, observed uncontrollables, and most importantly, enabled uncontrollables that have not yet been observed. We can transform the original WPSTNU W into a new WPSTNU W' by enforcing all the constraints on controllable and uncontrollable timepoints arising from X, and renormalizing the probability distributions $P(\Omega_i)$ on unobserved uncontrollables. Once this is done, we can build the new schedule s_t for W'. We will denote by Ω_t the cross product of uncontrollable outcomes remaining unobserved after t with probability distribution $P(\Omega_t)$. We can now evaluate the *true* expected value $g(s) = \int_{\omega \in \Omega_t} f(s, \omega) P(\omega_t)$ of the suffix of the original schedule s, and similarly the true expected value of the new schedule $g(s_t)$, and choose the better of the two schedules.

The selection criterion χ that implements the process described above is:

$$\chi_{EV}(s, s_t) = \begin{cases} s_t & \text{if } g(s_t) > g(s), \\ s & \text{otherwise.} \end{cases}$$

Computing tails of normal and beta distributions is all that is required to compute the expected value; this is not computationally expensive, and computing the true expected value ensures the selection criterion always chooses the best plan. Since our rescheduling policy π_t adheres to schedule s up to time t, and then either sticks with s or switches to a schedule with a higher expected value conditioned on X for an arbitrary set of uncontrollable outcomes $\omega \in \Omega_t$, it must be that

$g(\mathcal{R}) \geq g(s)$. The same argument can be extended to say that a policy π that reschedules arbitrarily many times, always checking to ensure the true expected value of the new schedule improves on the recomputed true expected value of the current schedule on the new problem W' before switching, cannot lead to a reduction in expected value relative to the initial schedule s. This proves the following result:

Proposition 1. *Let W be a WPSTNU and let s be a schedule for W. Let π be a rescheduling policy. Let χ_{EV} be the selection criterion defined above. Let $\mathcal{R} = (M, \pi, \chi_{EV})$ be a rescheduling approach for W. Then $g(\mathcal{R}) \geq g(s)$.*

Notice Proposition 1 is very general. *Any* rescheduling approach using χ_{EV} is guaranteed to ensure improvement in expected value, not just the fixed-period rescheduling approaches we focus on in this paper.

Algorithm 1: Simulated Rescheduling Approaches

Input : WPSTNU W
Input : Rescheduling approach $\mathcal{R} = (M, \pi_{Y,k}, \chi_{EV})$ ▷ MILP rescheduler M, event type $Y \in \{T, R, \Gamma\}$, rescheduling period k, selection criterion χ_{EV}
Var : Schedule s
Var : MILP representation P^W of W
$R' \leftarrow R$; ▷ Copy R to track observed uncontrollables during execution
$A' \leftarrow A$; ▷ Copy A to track executed controllables during execution
$T' \leftarrow A' \cup R'$; ▷ Tracking executed and occurred timepoints
$v \leftarrow \texttt{Realize}(R)$; ▷ Randomly generate uncontrollable durations
$t \leftarrow 0$; ▷ Placeholder for current time
$\tau_k \leftarrow \texttt{NULL}$; ▷ Placeholder for current timepoint
$P^{W'} \leftarrow P^W$; ▷ Placeholder for progressed MILP
$\Omega_t \leftarrow \times_i \Omega_i$; ▷ Placeholder for renormalized joint probability dist.
$s \leftarrow \texttt{Solve}(P^{W'})$; ▷ Generate initial optimal SC schedule
while $T' \neq \emptyset$ **do**
  if $(X = T) \vee (X = R)$ **then** ▷ Rescheduling approaches $\pi_{T,k}$ and $\pi_{R,k}$
    $\tau_k \leftarrow \texttt{GetNextTimepoint}(t, s, v, \pi_{Y,k})$;
    $t \leftarrow \texttt{GetTime}(\tau_k, s, v)$;
  else
    $t \leftarrow t + k$; ▷ Rescheduling approaches $\pi_{\Gamma,k}$
  $(P^{W'}, A') \leftarrow \texttt{UpdateControllables}(P^{W'}, A', s, t)$;
  $(P^{W'}, R') \leftarrow \texttt{UpdateUncontrollables}(P^{W'}, R', s, v, t)$;
  $(P^{W'}, \Omega_t) \leftarrow \texttt{UpdateDists}(P^{W'}, R - R', s, v, t)$;
  $s_t \leftarrow \texttt{Solve}(P^{W'})$;
  if $g(s_t) > g(s)$ **then** ▷ Follow selection criterion χ_{EV}
    $s \leftarrow s_t$;
  $(P^{W'}, A') \leftarrow \texttt{UpdateCurrentControllables}(A', s, t)$;
  $T' \leftarrow A' \cup R'$;
return s ; ▷ The 'as-executed' schedule

7.3 Rescheduling to Improve Expected Value

To implement our rescheduling approaches, we will rebuild the MILP for W' conditioned on X and re-solve it, to produce our proposed new schedule s_t. The process for updating the MILP by incorporating X is described in detail in the next section. Typically, the schedule can be improved after incorporation of constraints derived from X, so we expect the new MILP solution to reflect this improvement. Since the objective value of our MILP is a strong proxy for expected value, as shown in §6.4, we expect s_t will nearly always be chosen over s, except in rare cases where the under-approximation error for s_t so far exceeds that of s that the relationship between their MILP objective values is opposite that of their expected values. Nevertheless, we still explicitly evaluate the true expected value of the schedule resulting from the MILP solution in accordance with χ_{EV}.

Algorithm 1 shows our process for simulating rescheduling approaches for WPSTNUs. We compute the realization of all probabilistic durations up front but only use their values at the time of their realization. Contingent constraints cannot be accurately simulated since they lack probabilistic information of how nature chooses durations during execution. In order to simulate instances with contingent constraints, they are first converted into requirement constraints using Lemma 2. Therefore, a contingent constraint is only satisfied during simulation if the timepoints satisfy every outcome for the duration of the contingent constraint.

After initialization, the algorithm determines how much time passes before rescheduling occurs. To simulate event-based approaches, function `GetNextTimepoint` exploits the up-front generation of uncontrollable event times, and returns the k^{th} timepoint event into the future, and function `GetTime` returns the time this timepoint is either scheduled or occurs. Simulating time-based approaches simpler, requiring only extraction of k from the rescheduling policy description $\pi_{Y,k}$, and subsequent arithmetic.

The two functions `UpdateUncontrollables` and `UpdateControllables` each make various modifications to the MILP to incorporate the passage of time prior to rescheduling by solving the updated MILP. They also update the sets of timepoints (A', R') that have been executed or occurred, to drive the outer loop of the rescheduling simulation.

`UpdateControllables` updates the MILP by fixing the time of executed controllable timepoints and preventing unexecuted timepoints from being scheduled in the past. It does this by constraining the value of the MILP variable corresponding to a controllable timepoint a_i to its execution time $s(a_i)$ in schedule s. This operation is performed on each controllable timepoint that was executed since the previous reschedule, except for those scheduled to execute at t, allowing these timepoints to be rescheduled to a time in the future, as described in §7. `UpdateControllables` also sets the lower bounds of each MILP variable corresponding to the remaining unexecuted controllable timepoints to the current time t in order to prevent subsequent MILP solutions from rescheduling these timepoints to a time earlier than t.

`UpdateUncontrollables` updates components of the MILP corresponding to AR constraints $c(r_i, a_j)$ whose uncontrollable timepoint r_i has been observed (i.e., $v(r_i) = \tau \leq t$). We split C_u into two sets. C_u^{res} denotes the set of 'resolved' AR constraints (i.e., AR constraints $c(r_i, a_j)$ whose uncontrollable timepoint r_i was observed at or prior to time t, essentially transforming these constraints into rejectable STN constraints). C_u^{pend} denotes those that are enabled (i.e., r_i has not yet realized). For $c \in C_u^{res}$, the variable λ_{ij} is dropped from the MILP, and constraints 6, 7, 8 and 9 for $c(r_i, a_j)$ are removed and replaced with constraints

$$a_j - \tau \geq l_{r_i,a_j} - M(1 - \xi_{ij}),$$

$$a_j - \tau \leq u_{r_i,a_j} + M(1 - \xi_{ij}).$$

Note that the binary variable ξ_{ij} has now been re-purposed as a 'rejection decision' variable (similar to variables β_{ij}) for constraints in C_u^{res}; it can only take value 1 when $l_{r_i,a_j} \leq s(a_j) - \tau \leq u_{r_i,a_j}$. Note also that the left-hand sides of the added constraints reduce to a constant when both a_j has executed and r_i occurs prior to time t. We must also augment the objective function so that we obtain value $q_c(r_i, a_j)$ when constraints in C_u^{res} are known to be satisfied (i.e., $\xi_{ij} = 1$) by a proposed schedule:

$$\max \sum_{C_r} \beta_{ij} \mathbf{q_c}(\mathbf{a_i}, \mathbf{a_j}) + \sum_{C_u^{res}} \xi_{ij} \mathbf{q_c}(\mathbf{r_i}, \mathbf{a_j}) + \sum_{C_u^{pend}} \lambda_{ij} \mathbf{q_c}(\mathbf{r_i}, \mathbf{a_j}).$$

`UpdateDists` modifies the MILP by recomputing probability distributions on contingent edges and updating coefficients in the MILP corresponding to uncontrollable timepoints that have not been observed by time t. If a probabilistic duration constraint $d(a_i, r_i)$ has been activated (i.e. a_i has been executed) prior to time t but r_i has not yet occurred, then its probability distribution can be bounded below by t. This requires truncating and scaling the PDF of Ω_i to

$$P_i^t = \begin{cases} \dfrac{P_i}{1 - F_{ij}(t - s(a_i))} & \text{if } \Omega_i \geq t - s(a_i), \\ 0 & \text{otherwise;} \end{cases}$$

where F is the CDF of Ω_i. With the PDF modified in this way, we rebuild and re-approximate the functions F_{ij} (or F_{ji}), now denoting them as F_{ij}^t, for any AR constraints involving r_i, and update the $\overline{\boldsymbol{\alpha}}$, $\underline{\boldsymbol{\alpha}}$, $\boldsymbol{m}_{ij}^k$, and $\boldsymbol{c}_{ij}^k$ values used in MILP constraints 6, 7, and 9 as described in §5.1. Constraints 8 and 9 are updated to be bound above by this new piecewise linear approximation of F_{ij}^t.

Recall from Equation 3 in §5.1 that we can write the expected value of a fixed schedule s as

$$g(s) = \sum_{c \in C_s} \sigma(c(a_i, a_j), s) q_c(a_i, a_j) + \sum_{c \in C_u} F_{ij}(s(a_j) - s(a_i)) q_c(r_i, a_j).$$

When rescheduling at time t, we can represent the expected value of the original solution s conditioned on the updated distribution Ω_t relating to uncontrollable timepoints that we have observed up to time t, denoted $E(P(\Omega_t) f(s, \Omega_t))$, as

$$g(s_t) = \sum_{c \in C_s} \sigma(c(a_i, a_j), s) q_c(a_i, a_j) + \sum_{c \in C_u^{res}} \sigma(c(r_i, a_j), s, \omega) q_c(r_i, a_j) + \sum_{c \in C_u^{pend}} F_{ij}^t(s(a_j) - s(a_i)) q_c(r_i, a_j),$$

where C_u^{res} and C_u^{pend} are defined as above, using the renormalized PDFs. This is now our new objective function. Once all of these updates have been applied, we re-solve the MILP, obtaining a schedule s_t that may reschedule controllable timepoints scheduled to execute at or after time t, and compare its expected value to that of the old schedule s.

As described above, `UpdateControllables` does not fix the value of variables representing controllables that could execute at t, to allow the possibility of rescheduling. Once the next solution is chosen, `UpdateCurrentControllables` fixes all variables executing at t in the MILP. This routine constrains each controllable timepoint a_i for which $s(a_i) = t$ to ensure they are fixed in the MILP. While this eventuality is very unlikely for time-based approaches, it is not zero, and increases with rescheduling period. It is more important to protect against this scenario in event-based approaches, since rescheduling is triggered by execution or occurrence of timepoints, and we can't discount the possibility that multiple timepoints are scheduled to execute simultaneously.

7.4 Evaluating Rescheduling Approaches

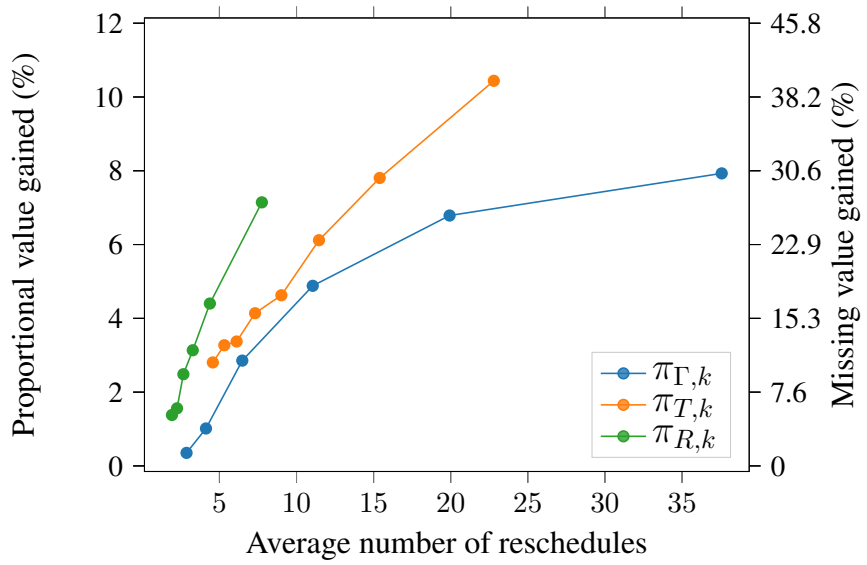
Derived Benchmark	Origin benchmark	Distribution	Added constraints
5	HEATlab	Normal	0
9	ROVERS	Normal (two values of σ)	5
10	ROVERS	Beta$(6, 1.5)$	5
11	ROVERS	Beta$(1.5, 6)$	5

Figure 16: The parameters of the derived benchmarks used for evaluating rescheduling policies. All of these derived benchmarks use value scheme 1 (intra-agent constraints have value 1, all inter-agent constraints, original and new, have value 5). All added constraints are inter-agent concurrency constraints. Each instance has a fixed makespan constraint of 50%. As discussed in §7.4, two different standard deviations σ are used in variants of benchmark 9.

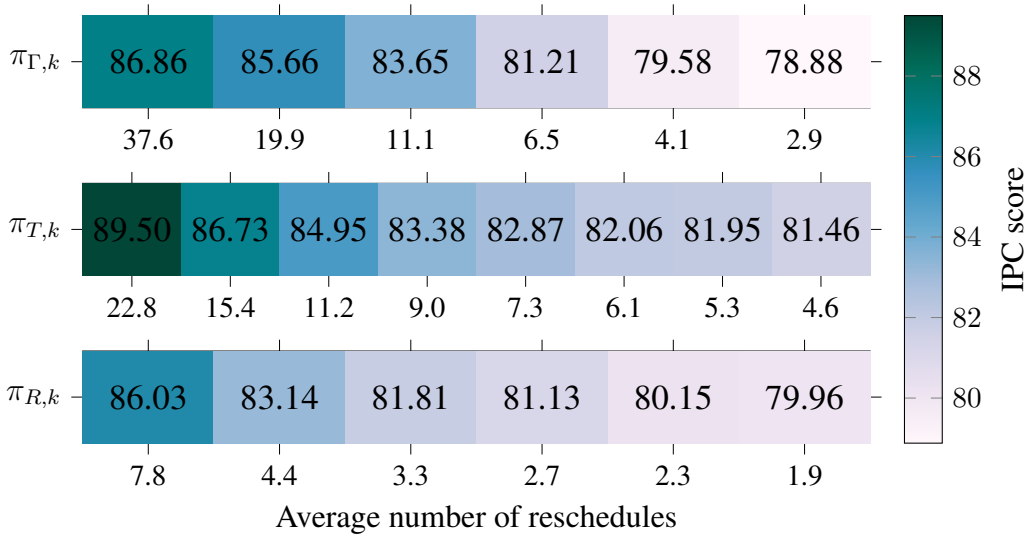
In order to empirically evaluate the benefits of rescheduling, we use both the MIT ROVERS benchmark introduced earlier, and a benchmark from the derived from the HMC HEATlab. Both benchmarks are similar inasmuch as they consist of multiple agents (rovers or robots) with both inter- and intra-agent constraints, which means we can use the same value schemes we used for the ROVERS instances to add different constraint values. The benchmarks are different enough that using both provides a good demonstration of the value of rescheduling across different types of problems. The HEATlab instances' time horizon is smaller than ROVERS instances. HEATlab instances have no contingent constraints, meaning they are PSTNs, which we turn into WPSTNUs. As we saw in the analysis in §6.4, the optimal fixed schedule for ROVERS instances without additional constraints or reduced makespan is quite close to the maximum possible value, thus providing little leverage for rescheduling. To make the ROVERS benchmarks more challenging and emphasize the value of rescheduling, we reduced the makespan by 50% and added a handful of rejectable inter-agent constraints. The HEATlab instances, by contrast, have more room for improvement due to rescheduling, but to maintain consistency of the benchmarks, we also reduce the makespan by 50% for the HEATlab instances. Finally, the HEATlab instances have chained uncontrollable durations, making it difficult to evaluate the implications of beta distributions on the results, since beta distributions are not closed under summation. Thus, we only evaluate HEATlab instances using normal distributions. Beta distributions in the rescheduling derived benchmarks were constructed in the same way as the earlier derived benchmarks. We summarize our rescheduling approach derived benchmarks in Figure 16.

We previously described some differences between the ROVERS and HEATlab benchmarks in §6. Before describing the empirical evaluation of rescheduling approaches, we point out another difference. In the HEATlab benchmarks, we note some of the original AR constraints may take the form $c(r_i, a_j) = [-y, 0]$, that is, a_j must *preemptively* be executed before r_i or immediately on observing the uncontrollable. The AR constraints in the ROVERS instances all take the form $c(r_i, a_j) = [y, x]$ with $0 \leq y < x$ This difference between the two benchmarks will become important when we discuss our results on rescheduling approaches below.

To quantify the benefit of rescheduling, we need to know how the initial MILP solution compares to the total value of all constraints in the initial problem, and how the rescheduling policy solution compares to the total value and the initial MILP solution. For WPSTNU P_e, denote $Q_t = \sum_{c \in C} q_c(t_i, t_j)$. If $\hat{B}$ is the value of the optimal MILP solution, then $Q - \hat{B}$ is the 'miss-



(a) The relationship between the average number of reschedules and the proportional value gained by each rescheduling policy.



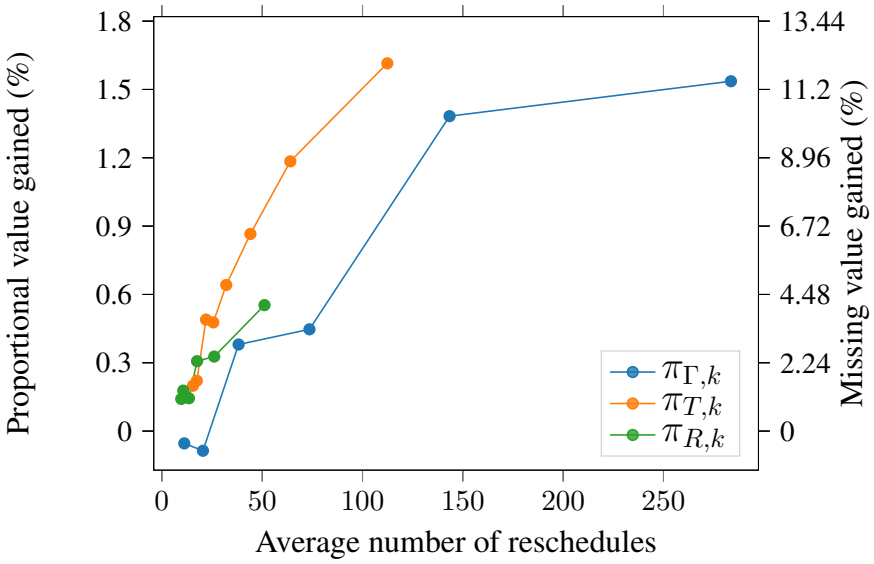
(b) The IPC scores comparing the performance of different rescheduling policies (in colored box). The value below the IPC scores show the average number of reschedules. See the beginning of §7.4 for how the IPC scores are computed.

Figure 17: Experimental results on derived benchmark 5 (HEATlab). Each problem instance is sampled twenty times for every rescheduling policy and the resulting value after execution is averaged. Rescheduling at all timepoints achieves a greater average value than rescheduling at clock ticks with fewer average reschedules. Rescheduling at only uncontrollable timepoints achieves similar average value to the most frequent clock tick rescheduling policy with far fewer average reschedules.

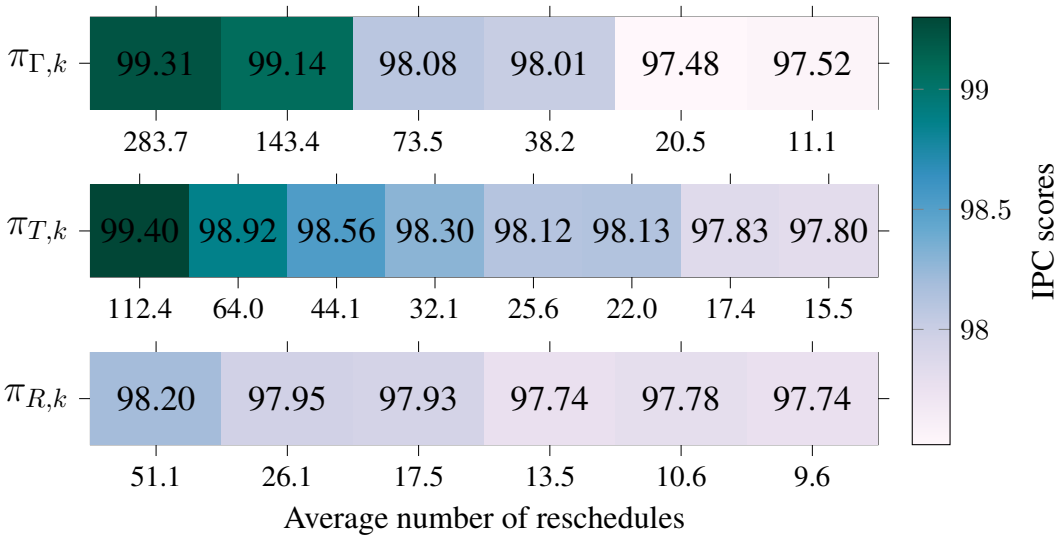
ing' value that could be regained by rescheduling. Let $\hat{B}_\pi$ denote the value achieved by a given rescheduling policy π. Then we can evaluate performance of π by examining two metrics. First, $\frac{\hat{B}_\pi - \hat{B}}{Q_t - \hat{B}}$ is the *missing value gained* by π. This value only measures improvement achieved by π in 'relative' terms. Specifically, if the missing value is very small, a large or small 'buy-back' is almost irrelevant. Second, $\frac{\hat{B}_\pi - \hat{B}}{Q_t}$ is the *proportional value gained*; this is the proportion of total value Q_t that is achieved by π and not the original MILP. This value measures the improvement in 'absolute' terms. By evaluating both metrics together, we can assess how useful rescheduling is versus the MILP solution alone; if there is a large 'buy-back' potential and rescheduling is able to restore a significant percentage of this value, that is a good result for rescheduling. Notice the two metrics differ only in the denominator.

Figure 17a and Figure 17b compare the performance of rescheduling approaches on derived benchmark 5, for the HEATlab instances using normal distributions. Each instance is solved and rescheduled 20 times, and the values in the plot represent averages over all instances in the benchmark. The x axis of both figures is the average number of reschedules. Because the number of event-based reschedules may vary, we use this axis instead of the rescheduling period k. For Figure 17a, the left y axis is the proportional value gained, or $\frac{\hat{B}_\pi - \hat{B}}{Q - \hat{B}}$, and the right y axis is missing value gained, or $\frac{\hat{B}_\pi - \hat{B}}{Q}$. Recall $\pi_{\Gamma,k}$ are time-based approaches, $\pi_{T,k}$ reschedules after any event occurs, and $\pi_{R,k}$ only reschedule when an uncontrollable event occurs. We first observe that the time-based approaches exhibit a point of diminishing returns with more reschedules. We also note event-based approaches generally reschedule less than the time-based approaches; this is understandable when we realize that there are only $|T|$ events, so the event-based approaches only reschedule approximately $|T|$ times, while the time-based approaches can reschedule much more often. Somewhat surprisingly, we observe that the event-based approach using policy $\pi_{T,k}$ achieves the best performance with fewer reschedules than second-place, and worse performing, $\pi_{\Gamma,k}$. We see that, compared to the SC schedule, up to 38.4% of the missing value can be restored by $\pi_{\Gamma,k}$, and this translates to roughly 11% of the total achievable value. The other approaches restore less than 30% of the missing value, or 7.5% of the total value.

Figure 17b shows the International Planning Competition (IPC) scores of the different rescheduling approaches. The IPC score is used to determine the best overall approach, as follows: for instance i, let M_i represent the proportion of total value obtained by the best approach. Then, for a given approach Y that achieves a proportion m_Y^i of total value on instance i, we calculate Score(i, Y) = $\frac{m_Y^i}{M_i} \times 100\%$. The IPC scores show the average number of reschedules for each rescheduling period k for each rescheduling policy class we compared. Note that the x axis differs for each rescheduling approach family. The heat chart shows a darker color for the better score. The IPC score is sensitive to the relative improvement of expected values achieved by different algorithms on the same instance, allowing for a direct comparison of the algorithms that isn't as easily achieved by examining the results of Figure 17a. Performance is aggregated over all algorithms' performance on the same problem instance, and performance over all instances is also aggregated. When compared this way, we see that the policy $\pi_{T,k}$ with the smallest rescheduling period, and therefore the maximum number of reschedules, is best. Furthermore, it achieves this performance by performing about 60% of the reschedules performed by the second best policy $\pi_{\Gamma,k}$. Rescheduling approaches $\pi_{R,k}$ performed worse than the other two, but surprisingly well, given that rescheduling only was permitted after uncontrollable events were observed; nevertheless, $\pi_{T,k}$ is preferable. We note that an IPC score of 100% indicates an algorithm that performs best on all instances. None of the reschedul-



(a) The relationship between the average number of reschedules and the proportional value gained by each rescheduling policy. Notice the significantly lower average value gained in comparison to the HEATlab data set.



(b) The IPC scores comparing the performance of different rescheduling policies (in colored box). The value below the IPC scores shows the average number of reschedules. See the beginning of §7.4 for how the IPC scores are computed.

Figure 18: Experimental results on derived benchmark 9 (ROVERS, Normal distributions). Each problem instance is sampled once for every rescheduling policy, and the resulting value after execution is averaged. Rescheduling at all timepoints achieves similar average value to rescheduling at clock ticks, and does so with fewer average reschedules.

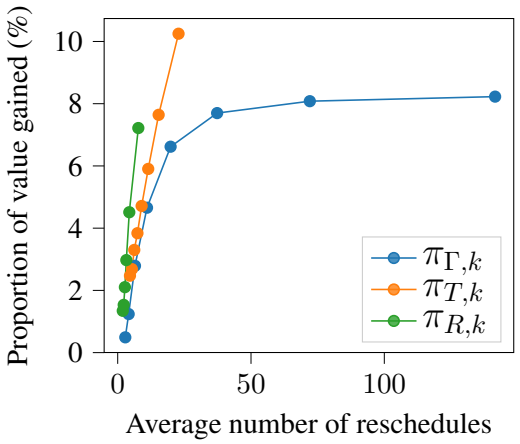
ing approaches exceeds an IPC score of 90% on the HEATlab instances, meaning no algorithm outperforms the others on all instances.

Figure 18a and Figure 18b compare the performance of rescheduling approaches on derived benchmark 9, for the ROVERS instances using normal distributions. Due to the large number of instances, only a subset was used to generate these plots, and each instance was solved once. As before, the x axis of Figure 18a is the average number of reschedules, the left y-axis shows proportional value gained, and the right y-axis shows the missing value gained. The results differ from those we observe for HEATlab instances. As we see from Figure 18a, only roughly 1.5% of the total value can be restored, and only roughly 11% of the missing value can be restored. We see the same diminishing returns from $\pi_{\Gamma,k}$ as rescheduling period decreases, but $\pi_{\Gamma,k}$ and $\pi_{T,k}$ appear close in performance at the maximum number of reschedules. Figure 18b shows the International Planning Competition (IPC) scores of the different rescheduling approaches for the ROVERS instances. We see a difference in algorithm performance compared to performance on the HEATlab instances; $\pi_{T,k}$ and $\pi_{\Gamma,k}$ with the smallest rescheduling period, and therefore the maximum number of reschedules, are almost indistinguishable, but $\pi_{T,k}$ achieves this performance with only 40% of the reschedules performed by $\pi_{\Gamma,k}$. We note all three algorithms achieve IPC scores of almost 100% when rescheduling as frequently as possible, indicating very little performance variation between approaches, but $\pi_{R,k}$ is a little worse than the other two approaches.

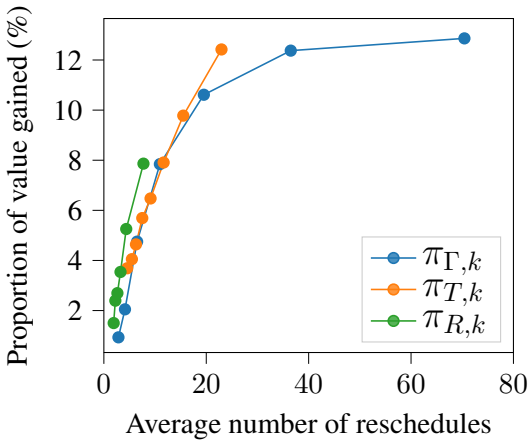
We now more closely examine the performance of rescheduling approaches on these benchmarks and answer the following questions: why is more frequent rescheduling needed for time-based approaches to match the performance of event-based approaches? Why is $\pi_{R,k}$ worse than $\pi_{T,k}$? And what accounts for the difference in performance of time-based approaches on ROVERS and HEATlab benchmarks?

We observe that performance of time-based approaches $\pi_{\Gamma,k}$ climbs, then tails off, with smaller k, and thus more rescheduling. We note instantaneous rescheduling upon event occurrences is almost impossible when using $\pi_{\Gamma,k}$, since times when rescheduling occurs aren't necessarily aligned with the scheduled execution of timepoints. Both time-based and event-based approaches can take advantage of observed uncontrollable timepoints, which resolve AR constraints into rejectable STN constraints. However, event-based approaches solve a less-constrained problem, because rescheduling occurs immediately after observing the uncontrollable. The difference is that the bounds of $c(r_i, a_j)$ are $[0, x]$ when rescheduling occurs instantly, while the bounds are $[y, x]$ for some $y > 0$, where y could be as high as the rescheduling period k. The longer the rescheduling period, the more constrained the problem is for a time-based approaches due to the lower bound on all unexecuted controllable timepoints. Frequent rescheduling is needed to respond to uncontrollable events in time to satisfy AR constraints and reduce this constraint penalty.

Given the large amount of information provided by observing an uncontrollable timepoint, why is $\pi_{R,k}$ worse than $\pi_{T,k}$? Waiting for the next uncontrollable to be observed may take too long and miss opportunities to update the schedule due to 'premature' commitment to satisfying a low-valued constraint. Consider a rejectable STN constraint $c(a_1, a_2)$ with bounds $[l_{a_1,a_2}, u_{a_1,a_2}]$ satisfied in schedule s such that $s(a_2) - s(a_1) = u_{a_1,a_2}$. Now consider an AR constraint $c(r_1, a_2)$ and suppose $v(r_2) > s(a_2)$. If r_1 is pending at $s(a_2)$, then it is possible that the renormalization of P_1 will make the expected value $\lambda_{12} q_c(r_1, a_2) > q_c(a_1, a_2)$, and thus postponing a_1 to increase the probability of satisfying $c(r_1, a_2)$, at the expense of rejecting $c(a_1, a_2)$, is worthwhile. Policy $\pi_{T,1}$ will reschedule at time $s(a_2)$, and policy $\pi_{R,1}$ will not reschedule at $s(a_2)$. This means $\pi_{R,1}$ will forego opportunities to improve the expected value.



(a) The relationship between the average number of reschedules and the proportional value gained by each rescheduling policy on derived benchmark 5. Same data as Figure 17a.



(b) The relationship between the average number of reschedules and the proportional value gained by each rescheduling policy on the modified version of derived benchmark 5.

Figure 19: A comparison between results on the standard version of benchmark 5 and a modified version where AR constraints $c(r_i, a_j) = [-y, 0]$) are flipped to be positive ($c(r_i, a_j) = [0, y]$). Shows that the performance of time-based rescheduling does not necessarily asymptotically approach the performance of event-based rescheduling when there are negative at-risk constraints.

Our results show that event-based approaches always beat the best time-based approaches on the HEATlab benchmark, but that both approaches yield comparable performance for ROVERS if time-based approaches reschedule very frequently. We previously noted that HEATlab instances contain AR constraints of the form $c(r_i, a_j) = [-y, 0]$. These are inter-agent constraints of high value $q_c(r_i, a_j) = 5$ per value scheme 1, and are often very tight (e.g. in some problems $c(r_i, a_j) = [-1, 0]$). Time-based rescheduling generally will only satisfy such constraints with low probability, because a_j must 'preempt' r_i. Furthermore, the expected value is small due to the tightness of the constraint bounds, making the expected value tradeoff unfavorable. By contrast, both event-based approaches can instantaneously reschedule a_j after observing r_i, and benefit from treating the AR constraint as a rejectable STN constraint of full (as opposed to expected) value, even though satisfying these constraints imposes a tight $[0, 0]$ constraint. We hypothesize the presence of these 'negative' AR constraints (of the form $c(r_i, a_j) = [-y, 0]$) explains the difference in performance between the time-based and event-based approaches. We performed a test on a modified version of derived benchmark 5, where the negative AR constraints' bounds were flipped to be positive ($c(r_i, a_j) = [0, y]$. Figure 19 compares the original performance of benchmark 5 and this modified version. Figure 19b shows that time-based approaches with small k, and thus more frequent rescheduling, and event-based approaches achieve similar expected values on this modified benchmark, suggesting that this feature of the HEATlab benchmark explains the difference in the performance of the approaches.

Figure 20 (spanning 2 pages) demonstrates the different performance on the ROVERS benchmark across derived benchmarks 9 through 11 as the skewness of distributions is varied. The first version of derived benchmark 9, described above, uses normal distributions for all probabilistic

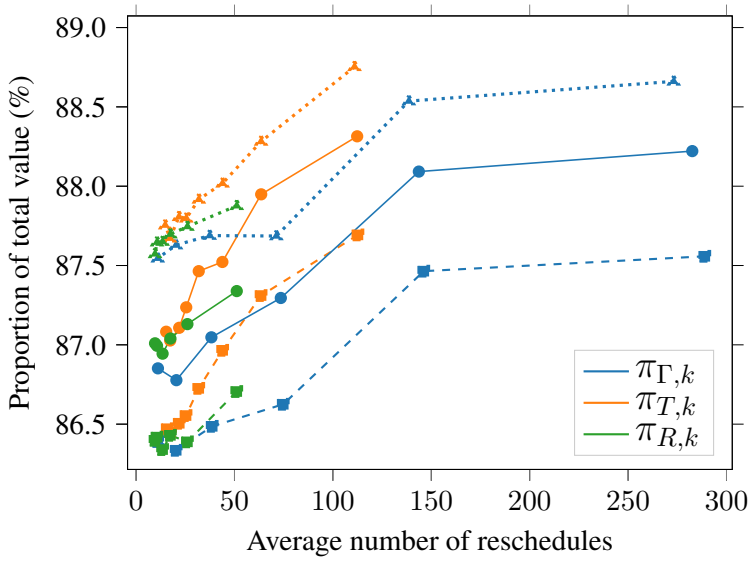
edges; in derived benchmarks 10 and 11, probabilistic edges with normal (μ, σ) distribution were converted into beta distributions with mean μ and lower and upper bounds $x - 2\sigma$ and $x + 2\sigma$, respectively, where the center x is chosen so that the beta distribution has mean μ. Since the support of our beta distributions is 4σ (rather than 1 like in a standardized beta distribution), the variance of the beta distribution is $\frac{(4\sigma)^2\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ where $\alpha = 1.5$ and $\beta = 6$. We also created a second version of benchmark 9, shown in Figures 20a and 20c, where the standard deviation of the new normal distribution is $\sqrt{\frac{(4\sigma)^2\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$. This was done to enable a head-to-head comparison of the performance of rescheduling on beta and normal ROVER instances with similar variances, but for which skewness drives short versus long uncertain durations as compared to the normal case. Notably, this means performance of rescheduling for ROVERS with normal distributions in Figure 20c differs slightly from that of Figures 18a and 20a.

Figure 20a aggregates the performance of normal and skew distributions on ROVER problems, showing the proportion of total value Q_t obtained by each scheduling policy on the different classes of distributions. As with the previous figures, we see the same pattern for each rescheduling policy class for each distribution. We see that skewed right distributions lead to better quality schedules than skewed left distributions, with the scaled normal distributions falling in between the two. The pattern of improvement of solution quality with increased rescheduling is more or less the same for each of the three rescheduling approaches; the proportion of total schedule quality achieved is almost linearly shifted up or down as a function of skewness. This suggests problem structure dictates buy-back potential more strongly than skewness, and that skewness biases the initial schedule value up or down.

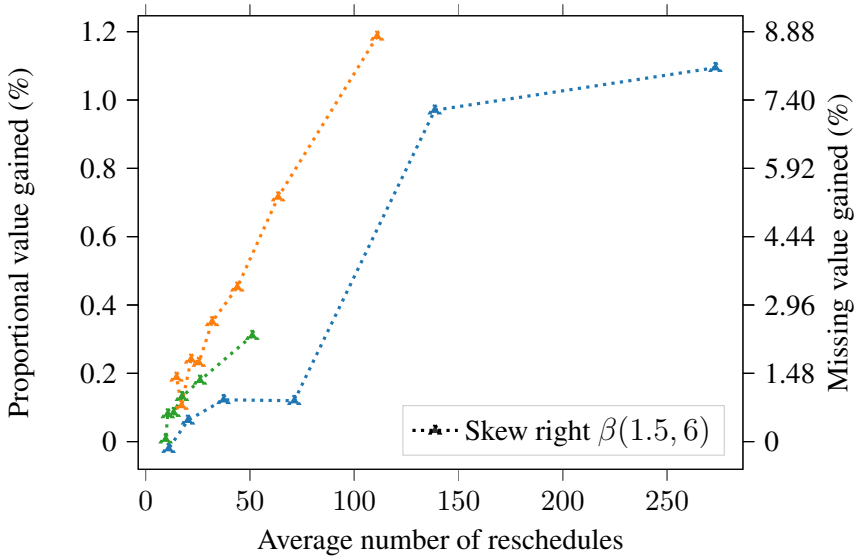
Figure 20b shows the performance of derived benchmark 11, which uses skew right beta distributions with parameters $\beta(1.5, 6)$. Figure 20c shows the performance of the second version derived benchmark 9 with the variance of normal distributions scaled such that they matched the variance of the associated skew left and right beta distributions, as described above. Figure 20d shows the performance of derived benchmark 9 which uses skew right beta distributions with parameters $\beta(6, 1.5)$. Each of these figures shows shows the proportional value gained on the left y-axis and the missing value gained on the right y-axis. The improved performance across different rescheduling approaches for beta distributions follows a similar relationship to that of the normal distributions. However, the figures indicate that the overall 'buy-back' potential of beta distributions is slightly lower than that of normal distributions, even when their variances are identical. The proportional value gained (left y-axis) for event-based scheduling in Figure 20c tops out at above 1.4% for normally distributed probabilistic durations, and is slightly lower for skew left (near 1.3%, Figure 20d) and even lower yet for skew right (near 1.2%, Figure20b.) The story is the same for missing value (right y-axis); Figure 20c tops out at around 9.6% for normally distributed durations, and is slightly lower for skew left (around 8.8%, Figure 20d) and even lower yet for skew right (around 8.1%, Figure20b.) This is consistent with the low potential of rescheduling to improve on the already high quality solutions for skew right case, and the ability of rescheduling to do better with normal distributions than with either of the beta distribution instances.

8. Conclusions and Future Work

When presented with a control problem on probabilistic simple temporal networks, the usual strategy of establishing strong controllability fails when constraints are guaranteed to be violated. To address this, we formally define a new type of temporal network, the Weighted Probabilistic Sim-

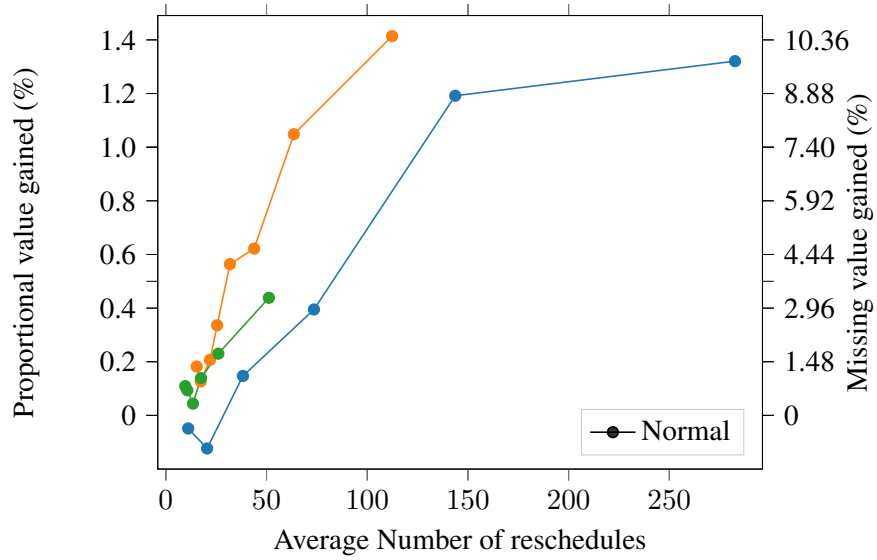


(a) The average proportion of value obtained for different rescheduling policies as a function of how many reschedules are performed on the ROVERS instances using derived problems 9 through 11. The solid lines represent derived benchmark 9 using scaled normal distributions, the dashed lines represent derived benchmark 10 with skew left beta distributions, and the dotted lines represent derived benchmark 11 with skew right beta distributions.

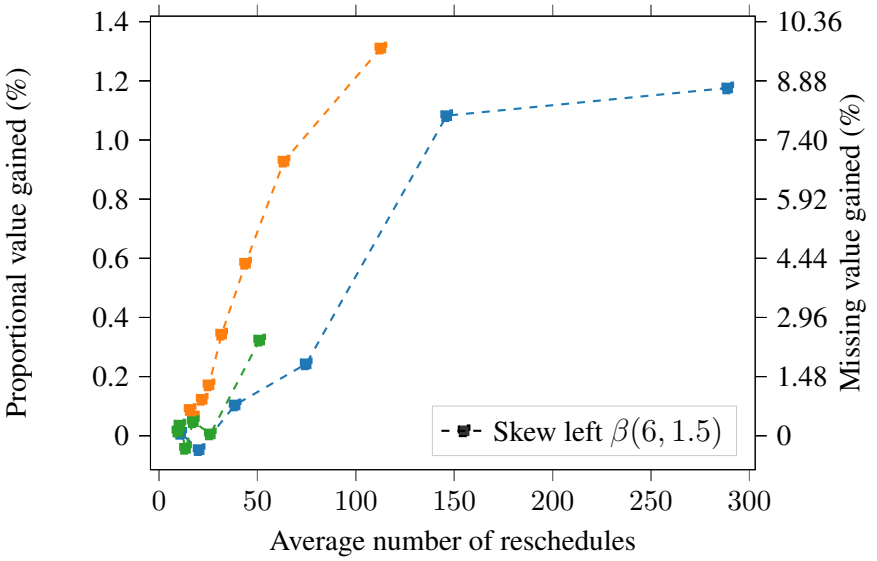


(b) The proportional value gained and the missing value gained for different rescheduling policies as a function of how many reschedules are performed on the ROVERS instances using derived benchmark 11. In this benchmark, probabilistic edges were altered to have skew right beta distributions with parameters $\beta(1.5, 6)$.

Figure 20



(c) The proportional value gained and missing value gained for different rescheduling policies as a function of how many reschedules are performed on the ROVERS instances using derived benchmark 9 modified such that normal distributions on probabilistic edges have standard deviation normalized to match the standard deviation of the associated beta distributions.



(d) The proportional value gained and missing value gained for different rescheduling policies as a function of how many reschedules are performed on the ROVERS instances using derived benchmark 10. In this benchmark, probabilistic edges were altered to have skew left beta distributions with parameters $\beta(6, 1.5)$.

Figure 20

ple Temporal Network with uncertainty (WPSTNU), and develop algorithms to solve the Expected Value Strong Controllability (EvSC) problem of finding a schedule maximizing the expected value of satisfied constraints on a WPSTNU. This formulation allows schedules to intentionally violate certain constraints in order to achieve maximum expected value during execution, providing a new way of handling 'over-constrained' scenarios for which risk-based formulations are not appropriate.

Solving the EvSC problem for WPSTNUs is a nonlinear optimization problem. We first formulate a MILP whose optimal solution bounds below the expected value of a WPSTNU. This MILP relies on a piecewise linear lower bound constructed for each AR constraint. Coupled with binary variables that control whether or not each rejectable constraint is satisfied, the MILP can balance the trade-off between satisfying STN constraints and covering probability mass (and therefore expected value) associated with AR constraints.

We first evaluate our MILP for EvSC using a series of modifications of the HEATlab and MIT ROVERS data sets. The modifications use different value schemes, added constraints, and reduced makespan, all of which are designed to exercise tradeoffs of expected value. Gurobi demonstrates good performance on this MILP. Problems without additional constraints are typically solved in under half a second; adding more constraints produces problems that are exponentially more difficult, as expected, but they are typically solved in under two minutes, which is comparable to previously demonstrated approaches. Reducing the makespan and adding more constraints generally forces tradeoffs between constraints for both benchmarks, as anticipated. However, HEATlab instances appear less impacted by the addition of more constraints, because they are proportionally less constraining, and therefore more easily satisfied. For ROVERS problems, the expected value remains quite high even when the makespan is reduced by 50%. For HEATlab problems, the expected value is initially more variable, and reducing the makespan has more impact on the expected value. Even though the MILP is guaranteed to bound below the expected value, empirical analysis of the error shows the MILP achieves almost optimal solutions, achieving an error of 10^{-6} for symmetric normal distributions for ROVERS instances and 10^{-4} for HEATlab instances. We also compare instances with normal and beta distributions, and show that while skewness impacts the quality of MILP solutions, as expected, it does not have a significant impact on the error of the solution.

We next introduce rescheduling approaches, which revise the schedule periodically after new information is gained. These approaches are characterized by an up-front fixed rescheduling period, and distinguished by the set of events or time ticks over which the rescheduling period is evaluated. Rescheduling approaches also require a selection criterion, which chooses to keep the current schedule or discard it in favor of the new schedule, by evaluating the expected value of both schedules. We prove that these rescheduling approaches always improve on the initial SC solution. We then empirically evaluate rescheduling approaches on the MIT ROVERS and HEATlab datasets, also varying the value schemes and distributions. We empirically quantify how well rescheduling improves the expected value, showing that the two benchmarks lead to different 'buyback' in expected value, but that more rescheduling is better, with 'diminishing returns'. Somewhat surprisingly, we find that rescheduling every k timepoint event occurrences (both controllable and uncontrollable) performs as well, or better, than rescheduling every k time units, with significantly less overall rescheduling effort. There are some differences in the relative merits of time-based rescheduling and event-based rescheduling on the HEATlab- and ROVERS-derived benchmarks. Event-based rescheduling dominates on the HEATlab instances, and rescheduling after uncontrollables only is competitive. However, on the ROVERS instances, time-based and event-based approaches have comparable performance, while uncontrollable-only performs poorly. We identify a

Benchmark	Dist.	Benchmark	Findings
1	Normal	ROVERS	SC achieves $> 95\%$ max value
2	Normal	ROVERS	SC achieves $85 - 93\%$ max value when makespan cut by 50%
3	Normal	ROVERS	SC achieves $> 80\%$ max value when adding constraints
2,3	Normal	ROVERS	More constraints, reduced makespan $\Rightarrow$ higher runtime
4	Beta	ROVERS	Left skew value $<$ right skew value
4	Beta	ROVERS	Skewness does not influence error
5	Normal	HEATlab	SC achieves lower, more variable % max value than ROVERS
6	Normal	HEATlab	SC achieves $40 - 90\%$ max value when makespan cut by 50%
7	Normal	HEATlab	SC achieves widely variable $\%$ max value when adding constraints
7	Normal	HEATlab	SC weakly increasing runtime when adding constraints
8	Normal	HEATlab	SC negligible impact of adding constraints
8	Normal	HEATlab	SC weakly increasing runtime of adding constraints
1-4	Both	ROVERS	Error is small (about 10^{-6})
5-8	Normal	HEATlab	Error is small (about 10^{-4})
5,9-11	Both	Both	Rescheduling improves value compared to SC
5,9-11	Both	Both	Event-based better than Time-based with fewer reschedules
5	Normal	HEATLab	HEATLab Time-based performance dependency (per Fig 13)
5	Normal	HEATlab	Regained 11% max value , 38.4% missing value
9	Normal	ROVERS	Regained 1.5% max value, 11% missing value (per Fig 18a)
10	Beta (left)	ROVERS	Regained 1.3% max value, 8.8% missing value
11	Beta (right)	ROVERS	Regained 1.2% max value, 8.1% missing value
10,11	Beta	ROVERS	Left skew value $<$ right skew value

Figure 21: A summary of the main empirical results.

feature of the HEATLAB instances, namely AR constraints requiring pre-emptive scheduling, that is present in HEATlab instances, but not in ROVERS instances, responsible for the difference in performance. Finally, we find that instances with skew right distributions lead to better quality than skew left distributions, that the 'buy-back' of skew distributions is somewhat smaller than that of normal distributions, and that there is a small difference in buy-back between skew left and skew right distributions, with skew left distributions providing slightly better buy-back. A summary of our main results is shown in Figure 21.

We conclude by describing several areas of future work: formally characterizing dynamic controllability, exploring different rescheduling approaches, further characterization of the properties of WPSTNUs, and generalizations of WPSTNUs.

While recent work (Akmal, Ammons, Li, Gao, Popowski, & Boerkoel, 2020) has resulted in algorithms for the dynamic control of PSTNs, the Expected Value Dynamic Controllability (EvDC) problem on WPSTNUs remains open. Solving this problem will provide executives with compact approaches similar to optimal policies for MDPs, and should lead to better expected value than the rescheduling approaches described in this paper. Algorithms to solve the EvDC problem are likely to be quite different than the techniques described in this work, and it is not clear how exactly to formulate, let alone solve, this problem. One intriguing line of research is to consider the EvDC problem as a finite-horizon MDP problem. The key challenge lies in defining the state space and state transitions. Doing so requires an up-front discretization of time; it is not obvious how to choose the discretization, and discretization may lead to a large (exponential) state space. Other options include extending (Gao, Popowski, & Boerkoel, 2020) or other approaches to the expected value case.

In lieu of solving the difficult EvDC problem, the fixed-period rescheduling approaches introduced in this paper can be improved upon in a variety of ways. Our event-based approaches can still miss opportunities to respond to uncertainty, and time-based approaches generated without regard to the initial schedule produced by the MILP miss opportunities to react to event occurrences. A hybrid algorithm that reschedules at a fixed time-based cadence after executing controllable timepoints may outperform the approaches described in this paper. Other approaches may compute the ideal rescheduling time directly.

We chose to explore a limited set of rescheduling approaches, using the true expected value as the foundation for our selection criterion. In particular, we note Proposition 1 does not, in fact, depend on the use of the MILP as the means to produce a new schedule, because the selection criterion uses the true expected value of a proposed new schedule s'. This proposition also doesn't require the optimal MILP solution as a starting point either, for the same reason, nor does it depend on a periodic rescheduling policy. In fact, the proposition doesn't even depend on the production of a single schedule when rescheduling is invoked; an arbitrary number of schedules could be generated, and *any* schedule whose expected value exceeds the current schedule could be selected. This suggests that many rescheduling approaches, using different methods to generate the initial schedule and proposed changes, still satisfy Proposition 1; the sole requirement is to check the true expected value of the old and new schedules.

We can also explore approaches that fall outside the confines of Proposition 1. For instance, can we simply use the MILP solution, and its quality, in the selection criterion? Suppose s_t is an optimal solution to the MILP we construct from $P^{W\varepsilon}$. The old schedule s can be evaluated as a candidate solution to this new MILP; doing so requires checking the satisfaction of resolved AR constraints, discarding the old values of λ_{ij}, recomputing them using the new piecewise bounds of F_{ij}^t, and recomputing the MILP solution value. Denote this value by $\hat{B}_t$. Denote the value of the optimal solution s_t to the new MILP by $\hat{B}'$. As we describe in §5.3, the MILP bounds below the expected value, and so s_t has a (very small) likelihood of being a worse solution than s. We can protect against this possibility by comparing the quality of s_t to the new quality of s plus the error bound on the new solution, $\hat{\epsilon}(s_t)$. The selection criterion $\chi_{EV}(s, s_t)$, using the MILP solution but accounting for the error, chooses s_t if $\hat{B}' > \hat{B}_t + \hat{\epsilon}(s_t)$. Doing so ensures there is no question that rescheduling always uses the best of the two candidate schedules. We then see that

$$g(s') \geq \hat{B}' > \hat{B}_t + \hat{\epsilon}(s_t) \geq g(s) \geq \hat{B}_t.$$

While this selection criterion preserves a formal proof that expected value is improved upon compared to the fixed schedule, it requires some additional work to compute the error $\hat{\epsilon}(s_t)$, but eliminates the work performed by χ_{EV} to compute the true expected value. Alternately, we can simply not worry about computing the error and use criterion $\chi_m(s, s_t)$, which, chooses s_t if $\hat{B}' > \hat{B}_t$. This final criterion avoids the extra integration needed to compute the error, or the true expected value of either schedule, but is no longer provably able to improve on the fixed schedule expected value. Nevertheless, the error in the MILP is low enough that it should perform well in practice. Comparing the runtimes and schedule quality of these different selection criteria is an interesting area of future work.

The HEATlab and ROVERS problems exhibit some differences in the potential for rescheduling to improve the expected value. The structure of the problems plays a role in the potential for rescheduling to improve expected value. The low potential improvement for the ROVERS instances

could be increased by investigating the buy-back potential for derived benchmark 3 (with up to 50 more constraints) or 4 (reduced makespan) or by varying the value schemes. An investigation of the structural properties of WPSTNUs and other variants is a worthwhile line of research.

The simple case of scalar-value preferences $q_c(t_i, t_j)$ can be extended to preferences over intervals, as is done for Simple Temporal Problems with Preferences (STPPs) (Rossi et al., 2006). Extending our work to address preferences creates a problem similar to the relaxable cc-pSTP of (Fang et al., 2014), but unifies the treatment of the lower value of achieving relaxed constraints and expected value. While we believe much of the theory described in this paper can be reused, leading to a similar MILP formulation and complexity, in order to address this problem, the generalization remains to be explored.

Finally, we can generalize WPSTNUs to express the quality of satisfying a group of constraints, rather than just a single constraint. Generalizing our MILP formulation is simple for weights of STN constraints, by dint of introducing a $0-1$ variable that can only take on the value 1 if all constraints in a group are satisfied, via a simple linear inequality. However, if a single quality applies to multiple AR constraints, the probability of satisfying all such constraints is the product of their individual probabilities, which leads to a nonlinear optimization problem. It is also not clear how a single weight or value for satisfying a mix of STN and AR constraints should be modeled.

9. Acknowledgements

We thank the anonymous reviewers of previous versions of this work for their valuable comments. This work was funded by the NASA Advanced Exploration Systems Program.

References

Abrahams, J. R., Chu, D. A., Diehl, G., Knittel, M., Lin, J., Lloyd, L., Boerkoel, J., & Frank, J. (2019). Reducing the computational and communication overhead of robust agent rescheduling. In *Proceedings of the* 29^{th} *National Conference on Automated Planning and Scheduling*, pp. 3–12.

Akmal, S., Ammons, S., Li, H., Gao, M., Popowski, L., & Boerkoel, J. (2020). Quantifying controllability in temporal networks with uncertainty. *Artificial Intelligence*, *289*, 33 – 84.

Boyan, J., & Littman, M. (2000). Exact solutions to time-dependent MDPs. In *Advances in Neural Information Processing Systems*, pp. 1026–1032.

Brooks, J., Reed, E., Gruver, A., & Boerkoel, J. (2015). Robustness in probabilistic temporal planning. In *Proceedings of the* 29^{th} *National Conference on Artificial Intelligence*, pp. 3239 – 3246.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*, 61–94.

Fang, C., Yu, P., & Williams, B. (2014). Chance-constrained probabilistic simple temporal problems. In *Proceedings of the* 28^{th} *National Conference on Artificial Intelligence*, pp. 2264 – 2270.

Gao, M., Popowski, L., & Boerkoel, J. (2020). Dynamic control of probabilistic simple temporal networks. In *Proceedings of the* 34^{th} *National Conference on Artificial Intelligence*, pp. 9851–9858.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

Ibragimov, I. A. (1956). On the composition of unimodal distributions. *Teor. Veroyatnost. i Primenen.*, *1*(2), 283–288.

Lund, K., Dietrich, S., Chow, S., & Boerkoel, J. (2017). Robust execution of temporal plans. In *Proceedings of the* 31^{st} *National Conference on Artificial Intelligence*, pp. 3597 – 3604.

Muscettola, N., Morris, P., & Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In *Proceedings of the* 17^{th} *International Joint Conference on Artificial Intelligence*, pp. 494 – 499.

Peinter, B., Moffitt, M. D., & Pollack, M. E. (2005). Solving overconstrained disjunctive temporal problems with preferences. In *Proceedings of the* 15^{th} *International Conference on Automated Planning and Scheduling*, pp. 202–211.

Rossi, F., Venable, K. B., & Yorke-Smith, N. (2006). Uncertainty in soft temporal constraint problems: A general framework and controllability algorithms for the fuzzy case. *Journal of Artificial Intelligence Research*, *27*, 617–674.

Saint-Guillain, M., Vaquero, T., Agrawal, J., & Chien, S. (2020). Robustness computation of dynamic controllability in probabilistic temporal networks with ordinary distributions. In *Proceedings of the* 29^{th} *International Joint Conference on Artificial Intelligence*, pp. 4168 – 4176.

Santana, P., Vaquero, T., Toledo, C., Wang, A., & Williams, B. (2016). PARIS: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In *Proceedings of the* 30^{th} *National Conference on Artificial Intelligence*, pp. 267 – 275.

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

Vidal, T., & Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, *11*(1), 23 – 45.

Vidal, T., & Ghallab, M. (1996). Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *Proceedings of the* 12^{th} *European Conference on Artificial Intelligence*, pp. 48 – 54.

Wang, A. J., & Williams, B. (2015). Chance-constrained scheduling via conflict-directed risk allocation. In *Proceedings of the* 29^{th} *National Conference on Artificial Intelligence*, pp. 3620 – 3627.

Weld, D., & Mausam (2006). Probabilistic temporal planning with uncertain durations. In *Proceedings of the* 21^{st} *National Conference on Artificial Intelligence*, pp. 880 – 887.

Yu, P., Fang, C., & Williams, B. (2015). Resolving over-constrained probabilistic temporal problems through chance constraint relaxation. In *Proceedings of the* 29^{th} *National Conference on Artificial Intelligence*, pp. 3425 – 3431.

Yu, P., Williams, B., Fang, C., Cui, J., & Haslum, P. (2017). Resolving over-constrained temporal problems with uncertainty through conflict-directed relaxation. *Journal of Artificial Intelligence Research*, *60*, 425–490.