Journal of Artificial Intelligence Research 75 (2022) 1323-1372 Submitted 10/2022; published 12/2022

# Chance-constrained Static Schedules for Temporally Probabilistic Plans

**Cheng Fang** CFANG@MIT.EDU
**Andrew J. Wang** WANGAJ@MIT.EDU
**Brian C. Williams** WILLIAMS@MIT.EDU
*MIT Computer Science and Artificial Intelligence Laboratory*
*32 Vassar Street*
*Cambridge, MA 02139 USA*

## Abstract

Time management under uncertainty is essential to large scale projects. From space exploration to industrial production, there is a need to schedule and perform activities. given complex specifications on timing. In order to generate schedules that are robust to uncertainty in the duration of activities, prior work has focused on a problem framing that uses an interval-bounded uncertainty representation. However, such approaches are unable to take advantage of known probability distributions over duration.

In this paper we concentrate on a probabilistic formulation of temporal problems with uncertain duration, called the *probabilistic simple temporal problem*. As distributions often have an unbounded range of outcomes, we consider *chance-constrained* solutions, with guarantees on the probability of meeting temporal constraints. By considering distributions over uncertain duration, we are able to use risk as a resource, reason over the relative likelihood of outcomes, and derive higher utility solutions. We first demonstrate our approach by encoding the problem as a convex program. We then develop a more efficient hybrid algorithm whose parent solver generates risk allocations and whose child solver generates schedules for a particular risk allocation. The child is made efficient by leveraging existing interval-bounded scheduling algorithms, while the parent is made efficient by extracting conflicts over risk allocations. We perform numerical experiments to show the advantages of reasoning over probabilistic uncertainty, by comparing the utility of schedules generated with risk allocation against those generated from reasoning over bounded uncertainty. We also empirically show that solution time is greatly reduced by incorporating conflict-directed risk allocation.

## 1. Introduction

Scheduling problems are ubiquitous and often involve complex interdependencies between timing of events. One example is ocean exploration. For ocean field deployments to be successful, schedulers must reason about uncertainty in the duration of activities. Activities associated with traversals are uncertain due to actuation noise and slip, and science activities are uncertain due to the variability in the time needed to perform experiments and collect data. Probability distributions for these uncertain durations are often known, for example, from dynamics modeling in the case of traversals and from prior history in the case of science activities. One approach to managing uncertainty is to construct a schedule that is feasible for all possible outcomes, but this approach is not viable for a commonly occurring case

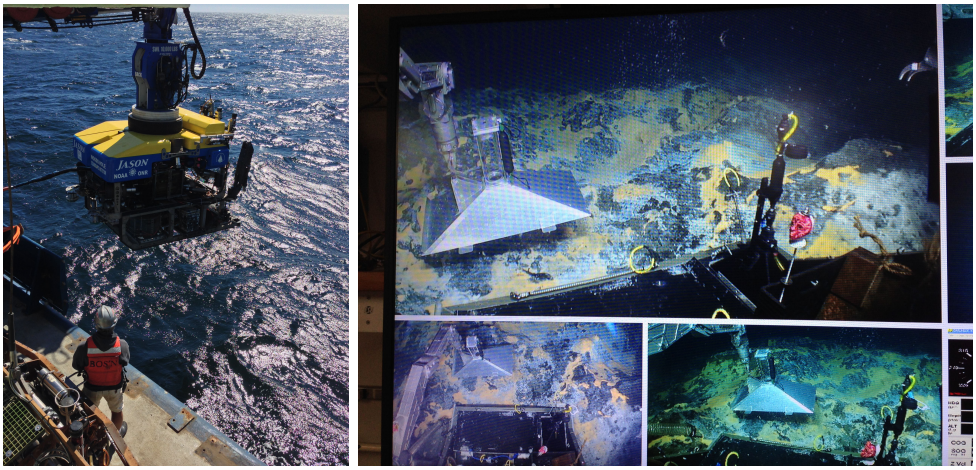


Figure 1: Oceanographic example. The Jason underwater vehicle operated by the Woods Hole Oceanographic Institute (left) is sent on missions to collect samples underwater, include water column samples and soil samples. The picture on the right shows the feed from onboard cameras during a water column sample mission.

in which the range of outcomes is unbounded, such as an uncertain position that behaves according to a normal distribution.

The following is an example of our alternative, of using a probabilistic problem statement.

**Example 1.** *Consider an oceanographic mission similar to that described in (Mendes et al., 2013), in which an unmanned vehicle and a crewed science vessel cooperate to gather and analyze samples from the seabed. The unmanned vehicle must travel to the site of interest and gather seabed soil samples before an eruption of a hydrocarbon plume. The scientists require at least 6 1/2 hrs for sample collection, while travelling to the site takes at least 4 hrs. The time of eruption is not known exactly before hand. However, given past observations, a distribution is available over eruption time relative to the start of day, with a mean of 15 hrs and a standard deviation of 2 1/2 hrs. Given this uncertainty, the scientists decide they will accept a risk of 5% of having less than 6 1/2 hrs to collect samples.*

In this example, the goal is captured by the requirement of having at least 6 1/2 hrs to collect samples. Importantly, it includes a corresponding upper bound on probability of failure of 5%, which acknowledges that there is risk, and specifies what risk is acceptable. Variables that influence this required collection time are time of eruption and travel time. Eruption is treated as a random variable and is described by a normal distribution, through a mean and variance. Travel time is treated as controllable and has a lower bound.

We refer back to this example to motivate our first contribution: *our introduction of the probabilistic simple temporal network (pSTN)*, for modeling scheduling problems with temporal uncertainty, and the definition of the chance-constrained probabilistic simple temporal problem (cc-pSTP), which asks for schedules that offer upper-bounds on the probability of requirement failure. This formulation allows us to describe uncertainties that arise from

many real world phenomena and activities, and allows us to find schedules that are of high utility with respect to an objective function, while meeting requirements on timing at a confidence level specified by the user.

It can be difficult to evaluate the probability of success of a particular schedule exactly. However, it is often easier to evaluate the probability mass covered by finite intervals over uncertain durations. This insight motivates the second major contribution, *an approximate encoding of cc-pSTP as a mathematical programming problem.* This encoding leverages prior results on scheduling with interval-bounded uncertainty, and incorporates the corresponding constraints within the mathematical program. The key to this contribution is a particular application of *risk allocation* (Ono and Williams, 2008) that allows us to come up with uncertainty sets that respect the chance constraints and result in feasible schedules with higher utility schedules.

Decomposition can often offer a more efficient alternative to framing a monolithic mathematical program. Our third contribution is a *decomposition of the encoding used to solve cc-pSTPs.* A parent sub-solver generates candidate uncertainty sets by allocating the risk, specified by the chance constraint, across the uncertain durations. A child sub-solver then checks schedulability of a candidate, using standard methods for problems with interval-bounded uncertain durations. A key to this contribution is to focus subsequent candidate search using conflicts discovered when a candidate is found infeasible. A conflict is a constraint denoting infeasible candidates (Stallman and Sussman, 1977; De Kleer and Williams, 1989). Together, the parent avoids unpromising candidates while the child offers efficient checking procedures.

While the formulation of the pSTN and the approximate encoding of cc-pSTP were documented together (Fang et al., 2014), the decomposition approach was separately described (Wang and Williams, 2015). The two approaches have complementary features.

The nonlinear optimization encoding of the cc-pSTP included the specification of an objective function, allowing higher quality schedules. This is in the tradition of prior work on simple temporal problems with preference (Khatib et al., 2001) and simple temporal problems with preference and uncertainty (Yorke-Smith et al., 2003). This contrasts with the approach using decomposition, which did not incorporate an objective function.

On the other hand, as the numerical encoding did not exploit the availability of solution methods specific to temporal networks, the numerical encoding did not benefit from the efficiency gains of the decomposition approach. Our fourth contribution is *the unification of the decomposition approach with the numerical approach, allowing efficient optimal scheduling through a conflict driven algorithm.* We thus combine the best features of our previous work, and provide an approach which optimizes with respect to an objective function as our algorithm did in Fang et al. (2014), while retaining similar runtime performance to our algorithm in Wang and Williams (2015).

The rest of this paper is structured as follows: Section 2 provides a survey of methods for scheduling with temporal constraints, with a focus on stochastic scheduling. Section 3 then presents our new representation for temporal networks with probabilistic uncertainty and states the chance-constrained probabilistic simple temporal problem (cc-STP). Section 4 provides an approximate encoding of the cc-pSTP as a convex program, expanding on our treatment in Fang et al. (2014). Section 5 reviews the decomposition approach given in Wang and Williams (2015) to finding a satisfying solution to the approximate cc-pSTP, and

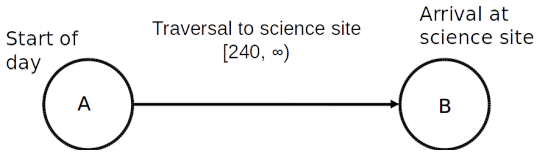


Figure 2: An STN for the traversal portion of the oceanography example described in Example 1. The two circles represent the controllable events and the solid line between represent the timing constraint between the events, specifying the allowed interval for the time difference between the two events.

discusses two methods for incorporating the objective function. Section 6 provides numerical results comparing the nonlinear optimization encoding approach, and the decomposition approaches, in terms of the quality of solution and the run time. Section 7 provides a summary of the key points of this work.

## 2. Background on Simple Temporal Networks

The problem of scheduling arises from temporal planning, where actions have duration and the start and end times of actions need to be determined. In temporal planning, not only must the generated sequence of actions be feasible, be enabled and reach the goal state, but the event times of this sequence must be consistent with a set of temporal requirements, such as deadlines and ordering constraints. The problem of scheduling events with stringent specifications is crucial to the success of science missions, shown in Example 1. As part of your plan, you need to come up with a schedule to meet these requirements.

*Temporal constraint networks* (Dechter et al., 1991), called temporal networks for short, offer a way to express the temporal structure of such plans. They consist of real-valued variables called *events* (sometimes called timepoints), which denote when actions start and end, and temporal constraints relating pairs of events. By drawing the events as vertices and the constraints as directed edges, a graph structure emerges, called a *temporal constraint network*. A network depicting the traversal portion of the oceanography example is shown in Figure 2. A scheduling problem involves finding an assignment to the events such that all temporal constraints are satisfied. Thus, a temporal network is often used to isolate the scheduling aspect of temporal planning. Uncertainty has been addressed before in temporal networks, but primarily in the form of interval bounds over possible values of uncertain variables (Vidal and Fargier, 1999; Venable and Yorke-Smith, 2005; Hunsberger et al., 2012). For example, an interval [5, 10] could specify that an action duration is guaranteed to be between 5 and 10 minutes, but its exact value can not be controlled. This is a simple but imprecise model of temporal uncertainty, as it is unable to express that certain durations are more likely than others; something that we can often extract from data or a model. To do so, we could replace interval bounds on durations with probability distributions, as was first introduced by Tsamardinos (2002). Using this as a starting point, our purpose is to revisit the problem statement of probabilistic scheduling and to offer an alternative class of problems and solution approaches, with particular attention to the concept of risk and risk bound.

Note that an important dimension of temporal network research, not explored in this paper, involves representing and reasoning about conditionality. Temporal constraints can be labeled by logical conditions, with these constraints activated only if their conditions evaluate to true, given the state of the world (Dechter et al., 1991; Stergiou and Koubarakis, 2000; Kim et al., 2001; Levine and Williams, 2018; Lau and Hoang, 2014). These conditions can represent controllable choices made during planning or uncontrollable effects observed during execution. Thus, conditional networks can express the temporal structure of plans with contingencies. When searching for a contingent plan, it is possible to leverage information from temporal constraints to prune the space of conditions (Conrad and Williams, 2011). However, this type of reasoning is largely orthogonal to our goal of realizing consistent schedules given uncertain durations. Therefore, this paper focuses on the *unconditional* setting, a class of networks called *simple temporal networks* (STNs), in which all constraints are always active.

The goal of the remainder of this section is to define the scheduling problem for an STN with probabilistic durations, including what constitutes a valid solution. To this end, we first establish the concepts of *consistency* and *execution*, drawing from the literature. First, we define STN and the concepts of scheduling and dispatching an STN. Then, we make some durations uncontrollable, with intervals bounding their uncontrolled values, thus forming a *STN with uncertainty* (STNU). Next we update the concepts of scheduling and dispatching accordingly. Building upon this background, in Section 3 we describe uncertain durations using distributions rather than intervals, thus creating a *probabilistic STN* (pSTN). Finally, we incorporate specifications of acceptable risk of failure by defining *chance constraints*, thus completing our new problem statement.

### 2.1 Simple Temporal Networks

Temporal constraint networks, as originally defined by Dechter et al. (1991), allow a constraint on the difference between two events to be expressed as a *disjunction* of intervals. For instance, Y may be allowed to happen either 1 to 3 time units or 5 to 9 time units after X, but not 3 to 5. In other words, $Y - X \in [1, 3] \bigcup [5, 9]$. This disjunction can be equivalently expressed as conditional constraints; intuitively, checking whether the disjunctive constraint is satisfied involves determining which interval is active. *Simple temporal networks* forbid multiple disjuncts outright (Definition 1).

**Definition 1.** *A* simple temporal network *(STN)* $\mathcal{N} = \langle E, C \rangle$ *is a constraint network consisting of*

- *real-valued* events $E$*, as its scope, and*
- simple temporal constraints (STCs) $C$ *on the duration between pairs of events* $X$ *and* $Y$*, of the form* $Y - X \in [l, u]$*, where* $X$ *and* $Y$ *are events, and* $l, u \in \mathbb{R}$*.* $Y - X$ *is called a* duration*.*

Events denote specific points in time. A simple temporal constraint may be equivalently expressed by the two constraints $l \leq Y - X \leq u$, denoting lower bound $l$ and upper bound $u$ on $Y - X$.

STNs are used to identify consistent *schedules* (Definition 2).

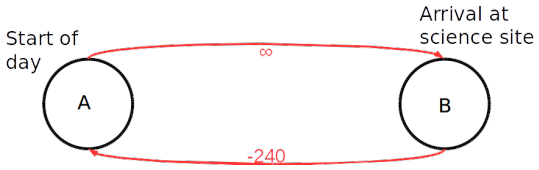


Figure 3: Distance Graph for the traversal portion of the oceanography example.

**Definition 2.** *A* schedule *of STN* $\mathcal{N} = \langle E, C \rangle$ *is an assignment to events* $E$ *that satisfies constraints* $C$*. A schedule of* $\mathcal{N}$ *is called a solution to a* simple temporal problem *with network* $\mathcal{N}$*. If a schedule exists, then the STN is said to be* consistent*.*

Because temporal constraints express bounds on the relative difference between pairs of events, a schedule can be offset forward or backward in time and still be valid. Hence, by convention, we typically introduce a *start event* $S$ that occurs at time 0 and precedes or co-occurs with all events, so that all events occur at non-negative times. A fundamental property of an STN is that it is consistent if and only if no negative cycles are contained in its equivalent *distance graph* (Dechter et al., 1991). A simplified STN and its distance graph for the Oceanographic example is shown in Figure 2 and Figure 3 respectively. To encode an STN with a distance graph, each event in $E$ is introduced as a vertex, and each constraint $l \leq Y - X \leq u$ in $C$ is expressed with two upper-bound directed edges, $X \xrightarrow{u} Y$ and $Y \xrightarrow{-l} X$, denoting constraints $Y - X \leq u$ and $X - Y \leq -l$, respectively. Each path through the distance graph represents an upper bound between the endpoints of that path, where the upper bound is the accumulated weights of the path. A *shortest path* between two events specifies the tightest constraint on their separation. Finally, a path that starts and ends on the same event $e$, that is, a cycle, denotes constraint $e - e \leq w$, where $w$ is the path weight. This constraint is inconsistent if and only if $w$ is negative.

### 2.2 Uncertainty and Controllability

In a standard STN formulation, simple temporal constraints are used to represent both temporal requirements and the duration of actions. An implicit assumption is that the duration of actions are fully controllable: like requirements, we may choose any value for action duration that lies within its bounds. However, this is insufficient to represent many real world scenarios, as actions may have uncontrollable durations. In this case we cannot choose the precise duration of the actions, but we may know their duration within given bounds. The *simple temporal network with uncertainty* models these scenarios (Vidal and Fargier, 1999) (Definition 3).

**Definition 3.** *A* simple temporal network with uncertainty *(STNU) is defined as* $N^u = \langle E^c, E^u, C^r, C^c \rangle$*, where*

- $E^c$ *is a set of real-valued* controllable events *that the agent can assign;*
- $E^u$ *is a set of real-valued* uncontrollable events *assigned by nature;*
- $C^r$ *is a set of* requirement constraints*,* $y - x \in [l, u]$*, for events* $x, y \in E^c \bigcup E^u$*,* $l, u \in \mathbb{R}$ *and*

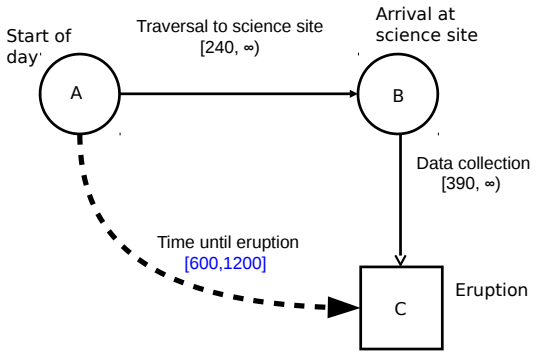


Figure 4: STNU representation of the oceanography example.

- $C^c$ *is a set of* contingent constraints*,* $Y - X \in [l, u]$ *where* $X \in E^c \bigcup E^u$*,* $y \in E^u$*, and* $l, u \in \mathbb{R}^+$*.* $Y$ *is called the* terminating event*. Every* $Y \in E^u$ *is a terminating event of exactly one* $c \in C^c$*.*

*A duration* $Y - X$ *is* controllable *if* $X, Y \in E^c$ *and* uncontrollable*, otherwise.*

A STNU differs from a STN in that events are divided into *controlled* and *uncontrolled* events. Likewise, simple temporal constraints are divided into *requirement* and *contingent* constraints. Both bound the difference between events to an interval. However, while requirement constraints allow their difference to be chosen by design, contingent constraints are used to represent actions with uncertain durations whose values are chosen by nature.

We might map the scenario from Example 1 to an STNU as follows.

**Example 2.** *Given the uncertainty surrounding the time of eruption, the oceanographers typically provide estimated upper and lower bounds on the uncertain duration to this event. Recall that the scientists set the probability of failure to complete sample collection before eruption to be less than 5%. The scientists decide to distribute this risk evenly and consider only the* $2\sigma$ *interval for the time of eruption, by assuming a normal distribution for the time of eruption. This results in the STNU depicted in Figure 4.*

It follows from this definition that there is a 1-1 mapping between uncontrolled events $E^u$ and contingent constraints $C^c$. A contingent constraint must terminate on an uncontrollable event. Conversely, every uncontrollable event has exactly one contingent constraint that terminates on it. The reason for this is that the duration associated with every contingent constraint corresponds to an independent stochastic process, hence it does not make sense for two contingent constraints to determine the value of the same event.

It is often the case that we would like to model two independent threads of activity that each have uncertain durations and are followed by a common event. We cannot guarantee that these two threads end at the same time, hence one thread may have to wait for the other thread to finish. We can model these waiting times using $[0, +\infty)$ requirement constraints.

We place no restrictions on the start event of an uncontrollable duration. It could be a controllable event that is scheduled by a policy, or it could be an uncontrollable event that ends a previous uncertain duration.

To characterize the behavior of an STNU for different values of uncontrolled events, we introduce the concept of *outcome* (Definition 4).

**Definition 4** (Outcomes and scheduling policies)**.** *An* outcome $\omega$ *of STNU* $N^u$ *is a set of assignments to the durations of its contingent constraints* $C^c$*.* $\omega_{X,Y}$ *denotes the assignment* $\omega(Y - X)$ *to duration* $Y - X$*.*

*A* scheduling policy $\pi : \mathbb{R}^{|C^c|} \rightarrow \mathbb{R}^{|E^c|}$ *for* $N^u$ *assigns controllable events* $E^c$ *based on outcomes* $\omega$ *for its contingent constraints* $C^c$*. A policy denotes a* fixed schedule *if* $\pi(\omega) = \pi(\omega')$ *for all outcomes* $\omega$ *and* $\omega'$*.*

In this paper we only consider outcomes as complete assignments to durations of contingent constraints. When an outcome $\omega$ assigns $\omega_{XY}$ to the duration of a contingent constraint $X \xrightarrow{[l,u]} Y$, that contingent constraint effectively becomes $X \xrightarrow{[\omega_{XY},\omega_{XY}]} Y$. There is thus no uncertainty in the contingent constraint, given the outcome, and the contingent constraint is equivalent to requirement constraint $[\omega_{XY}, \omega_{XY}]$. As a consequence, given an outcome, we may reduce an STNU to an equivalent STN, by replacing all contingent constraints with requirement constraints. This STN is called the *projection* $p_\omega$ of the STNU under outcome $\omega$.

The concept of correct fixed schedule with respect to possible outcomes is captured by *strong controllability* (Definition 5).

**Definition 5** (Strong Controllability)**.** *Consider STNU* $\mathcal{N}^u = \langle E^c, E^u, C^r, C^c \rangle$*. Let* $\Omega$ *be the set of outcomes for contingent constraints* $C^c$*. We say that* $\mathcal{N}$ *is* strongly controllable *if there exists a fixed schedule* $\pi$ *such that for any* $\omega \in \Omega$*, all constraints in* $C^r$ *are satisfied.*

Intuitively, an STNU is strongly controllable if there exists a schedule such that, for any outcome of its contingent constraints, its set of requirement constraints is satisfied. There is thus at least one schedule that always "works".

We illustrate strong controllability with the following example, which shows that the scenario given in Example 2 is not strongly controllable.

**Example 3.** *Consider again Figure 4, depicting an STNU with a* $2\sigma$ *interval around mean for its uncertain duration. Given the 240 minute lower bound for traversal time and the 390 minute lower bound required to collect data, we need at least 630 minutes to complete the mission. However, the contingent constraint specifies that the eruption could occur as early as 600 minutes. Thus, there is no schedule that guarantees for all eruption times that the data will be gathered before the eruption occurs. The STNU is thus* not *strongly controllable.*

One way to think about strong controllability for STNUs is to think in terms of individual outcomes. Recall that for every STNU outcome, we may replace the contingent constraint with requirement constraints. For the volcano survey example, consider the outcome in which contingent constraint AC is assigned 600. We could replace AC with a requirement constraint [600,600].

Since strong controllability requires a fixed schedule that works for all possible outcomes, one way to determine strong controllability is to examine all possible outcomes and see if

| Case | Conditions | Reduction |
|---|---|---|
| 1 | $x$ controllable, $y'$ uncontrollable with contingent constraint $y' - y \in [l_y, u_y]$, requirement constraint $y' - x \geq a$, | $y - x \geq a - l_y$ |
| 2 | $x$ controllable, $y'$ uncontrollable with contingent constraint $y' - y \in [l_y, u_y]$, requirement constraint $x - y' \geq a$ | $x - y \geq a + u_y$ |
| 3 | $x'$ uncontrollable with contingent constraint $x' - x \in [l_x, u_x]$, $y'$ uncontrollable with contingent constraint $y' - y \in [l_y, u_y]$, requirement constraint $y' - x' \geq a$ | $y - x \geq a - l_y + u_x$ |

Figure 5: Strong controllability reductions (Vidal and Fargier, 1999).

there is a consistent schedule across all outcomes. This is impractical, since contingent constraints are assigned real-values within intervals, and thus there is an uncountable number of outcomes, except degenerate cases.

The key observation made by the STNU literature is that we can represent the effects of uncertainty on the controllable events and durations of the network through derived requirements constraints. Intuitively these derived constraints eliminate all schedules that are inconsistent with *any* outcome, together with the original requirement constraints. If a schedule satisfies these derived constraints, it is guaranteed to be consistent with the STNU for any outcome.

A complete set of derived requirement constraints for an STNU is constructed by computing the closure of a set of strong controllability *reductions* (Vidal and Fargier, 1999) (Table 5). This closure can be viewed as eliminating each uncontrollable event through bucket elimination (Dechter, 1999). Together the reductions take each requirement constraint that includes an uncontrollable event and derives from it requirement constraints between controllable events. The net effect of this process is to reduce a STNU strong controllability problem to a STN consistency problem.

Performing the reductions over all constraints takes linear time (Vidal and Fargier, 1999) and results in an STN that largely resembles the original STNU, but stripped of uncontrollable events. With this reduction, the entire theory and set of algorithms for STNs can be ported over to strongly controllability of STNUs. This means that checking STNU strong controllability after reduction, is equivalent to checking STN consistency, which has $O(nm)$ complexity.

In the following example, we give a walk through of the strong controllability algorithm.

**Example 4.** *Consider again Figure 4. From edges $\overrightarrow{AC}$ and $\overrightarrow{BC}$ we use reduction Case 1 to derive a new edge $\overrightarrow{BA}$, as shown in Figure 6. Case 2 is too weak, since the upper bound is infinite, and Case 3 does not apply. Since all reductions have been applied, this new edge captures all constraints imposed by the STNU on controlled events A and B, allowing us to reduce the STNU to an equivalent STN over A and B. Note that the new edge $\overrightarrow{BA}$ can*

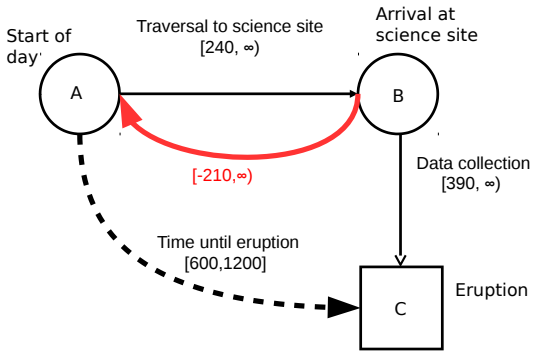


Figure 6: Strong controllability checking for the oceanography scenario.

*also be read as imposing an upper bound of 210 in the direction* $\overrightarrow{AB}$*. This conflicts with the original lower bound of 240 on the traversal. Hence the reduced STN is inconsistent, and the STNU is not strongly controllable.*

Note that our proof that the problem was not strongly controllable depended on the specified bound for the time of eruption. The alternative bound of $[630, 1400]$ would result in a strongly controllable STNU, while still covering a 95% probability mass. We will return to this observation when we present our probabilistic approach to scheduling under uncertainty, in which we consider the relative likelihood of outcomes.

## 3. Problem: Probabilistic Temporal Networks with Chance Constraints

The STNU formulation of uncertainty offers two powerful benefits. First, it is rooted in the common real world practice of using intervals to reason about uncertainty simply. Second a very rich theory and set of efficient algorithms have been developed for reasoning over this representation.

There are, however, two significant drawbacks. First, the criteria of strong controllability can be difficult to satisfy in practice. The reason is that a schedule violates strong controllability if it is inconsistent for even a single outcome, no matter how unlikely the set of violating outcomes might be. Second, coming up with appropriate intervals for contingent constraints is an art; the literature has not yet provided an objective criteria for constructing and evaluating the accuracy of contingent constraint intervals, relative to a particular real world situation.

To address these limitations we introduce a probabilistic problem formulation that associates a probability distribution, rather than an interval, with each contingent duration. This allows us to extract a probability for every outcome, rather than simply classifying each outcome as possible or not. We call this new model a *probabilistic Simple Temporal Network* (pSTN).

In addition, we acknowledge that for many real-world problems, risk of failure cannot be completely removed, and that many real-world processes are accepting of some level of

failure, if not too excessive. We model this by specifying bounds on the risk of failure to meet the requirement constraints, called *chance constraints* (Charnes and Cooper, 1959; Miller and Wagner, 1965) by the stochastic optimization community.

Alternative extensions of STNs with probabilistic durations have been introduced in the community. One such formulation (Tsamardinos, 2002) introduced stochasticity into the problem by considering the timing of uncontrollable events as random variables, with probability distributions conditioned on the choice of execution times for controllable events. This formulation was problematic, as the timing of controllable variables are not naturally cast as random variables.

The timing of controllable variables are influenced both by the choice of execution strategy and the timing of uncontrollable events. It was thus difficult to define the probability distribution of the timing for each controllable event. This is carried over to the conditional distributions for the uncontrollable variables, the construction of which required the probability distributions of the controllable variables. The work in Tsamardinos (2002) focused on static scheduling, and the controllable events could be scheduled regardless of the outcome of the uncontrollable events. While this may motivate the construction of probability distributions for controllable events as indicator functions, this would require the schedule of the controllable events to be known a priori.

A revision of this formalism partially addresses these concerns (Tsamardinos et al., 2003). While the absolute timing of the uncontrollable events are still defined as the random variables in the problem, the associated probability distributions are indirectly constructed using the time difference between uncontrollable events and controllable events. These time differences were conditioned on the timing of controllable events. However, the probability distributions were independent of the assignments to the controllable events. The key insight, which also motivates our formulation below, is that stochasticity is not introduced by the timing of the uncontrollable events. Rather, uncertainty in the problem is introduced due to the timing of the uncontrollable events *relative to* the timing of the controllable events. Our definition thus focuses on random variables describing the duration of the contingent constraints.

A related approach (Lund et al., 2017) considered maximizing the robustness of scheduling strategies to failure. In both cases, the decision making is performed without an objective function, potentially leading to conservative strategies that result in excessive cost. Our cc-pSTN formulation lends it self to optimizing an objective function that measures performance, while incorporating risk as a constraint. Alternatively, a robust optimization approach was explored in Lau et al. (2006), in which only the mean and variance of the random variables were considered. Rather than using only the mean and variance of the random variables, our approach uses the full distributional information in order to avoid conservatism.

There is also a branch of research into resolving over-constrained chance-constrained simple temporal problems (Cui et al., 2015; Yu et al., 2015, 2017). Similar to the work described in Tsamardinos (2002), the work focuses on the feasibility of solutions. Preference models are specified over the constraint, so that the solver may be able to restore feasibility by relaxing conflicting constraints. Like this work, the relaxation algorithm uses negative cycle detection to efficiently prove infeasibility and provide an concise representation of infeasibility. This is because both approaches rely on decompositions of the problem, in

which controllability is checked via reformulations to STNs. Negative cycles are natural summaries of conflicts, and are related to feasibility cuts in a Bender's Decomposition framework given a system of difference constraints (see Appendix B).

While preferences are also represented in our work in the form of objective function, they are not the focus of our work. Preference functions in the context of simple temporal networks were discussed with a semi-ring formalism (Khatib et al., 2001), a general framework which covers the makespan objective function used in our evaluations. The work was extended to deal with set-bounded uncertainty (Yorke-Smith et al., 2003), and we adopt a similar focus on optimality in the context of a fixed schedule in our work.

There has also been work on scheduling under uncertainty beyond the STN community. The cc-pSTN formulation can be considered a particular kind of stochastic constraint programming (Walsh, 2002; Hnich et al., 2012). However, the cc-pSTN can be thought of as chance-constrained version of a problem involving only difference constraints. This is a key feature that we exploit in Section 5 in deriving an efficient solution method for cc-pSTNs.

A related line of research in the constraint programming community focuses on stochastic jobshop scheduling (Beck and Wilson, 2007), as well as extensions such as resource-constrained project scheduling (Fu et al., 2012). The traditional jobshop scheduling focuses on the order of the jobs, rather than the time assignments, without specifications of temporal constraints, and thus does not naturally describe the same set of problems as STNs and extensions of STNs. The resource constrained extension is also restrictive, focusing on specifications of time differences between activities, rather than temporal constraints between time points in general.

Lastly, recent work has focused on quantifying the probability of feasibility given an execution strategy and probabilistic uncertainty in timing (Saint-Guillain et al., 2020, 2021). Rather than constructing a static schedule which meets guarantees on the probability of success as in this work, the authors have focused on computing the probability of success when an execution strategy has been specified. The alternative approach has the advantage of being applicable to dynamic strategies, whereas we confine ourselves to generating static schedules. The other key difference is that we assume that the uncertain timing is described with continuous distributions, while the prior work considers discrete distributions.

We start with the following definition of probabilistic STN (Definition 6). An example is provided in Figure 7.

**Definition 6.** $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$ *defines a* probabilistic simple temporal network *(pSTN), with*

- $E^c, E^u, C^c$ *as in Definition 3, and*
- $C^p$ *is the set of* probabilistic *contingent constraints. For each* $c_{XY} \in C^p$ *from* $X$ *to* $Y$*, there is a known probability distribution* $d$ *such that* $Y - X \sim d$*.*

Having made the uncontrollable durations probabilistic in a pSTN, we now consider what this means for controllability. Whereas the uncontrollable durations of an STNU are interval-bounded, the tails of pSTN distributions may have unbounded support. When scheduling pSTNs, we may need to consider more extreme outcomes that could preclude the scheduler from *always* producing valid event sequences, that is, satisfying all requirement constraints. The key, however, is that due to a well-defined joint distribution over outcomes,

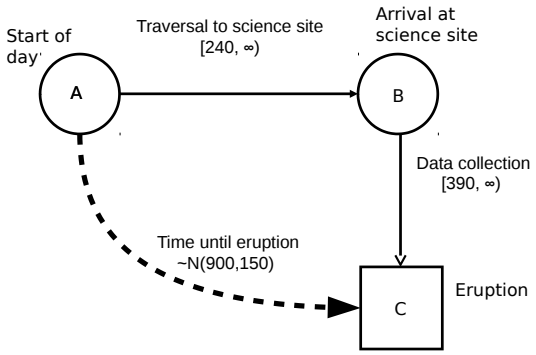


Figure 7: Oceanographic scheduling example.

we can compute success probability in terms of *the cumulative probability of outcomes* for which a policy's resulting schedule satisfies the requirement constraints. This motivates the following definition of risk for a scheduling policy (Definition 7).

**Definition 7** (pSTN policy risk)**.** *Consider pSTN* $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$*. For policy* $\pi$*, let* $\Omega_\pi \subseteq \Omega$ *be the subset of the sample space where some requirement constraint is violated, that is, for all* $\omega \in \Omega_\pi$*, there exists some constraint* $c \in C^r$*, where* $c : Y - X \in [l, u]$*, such that* $Y(\omega) - X(\omega) \notin [l, u]$*, then* $\omega \in \Omega_\pi$*.*

*The* risk *of policy* $\pi$ *with respect to the set of requirement constraints* $C^r$ *is, therefore,*

$$\rho(\pi) = P(\Omega_\pi).$$

This definition conceptually evaluates risk by integrating the probability measure over all outcomes for which a policy satisfies its requirements. To do so, the policy can be used to map each outcome to a schedule over controllable events and check this schedule against the requirement constraints for violation.

Using our definition of policy risk, we define the pSTN scheduling chance-constrained satisfaction problem as finding a policy whose risk is below an acceptable bound. The formal definition of the chance-constrained probabilistic simple temporal problem (cc-pSTP) is defined as follows.

**Problem 1** (Chance-constrained probabilistic simple temporal problem)**.** Given:

- $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$*, a pSTN; and*
- $\Delta \in [0, 1]$*, an upper bound on the risk of failure.*

Find:

- $\pi$*, a scheduling policy for* $E^c$ *based on outcomes of contingent constraint durations.*

Subject to:

- $\rho(\pi) \leq \Delta$, *the probability of policy risk being bounded by* $\Delta$.

$\mathcal{N}^p$ *is said to be* controllable with probability of failure at most $\Delta$ *if and only if such a policy* $\pi$ *exists.*

The above problem is defined in general terms, allowing for different types of scheduling policies. A dynamic policy, assigning controllable events as the outcomes of contingent constraints are observed, would be *dynamically* controllable with required probabilities of success.

In our subsequent treatment, we focus on static schedules for controllable events. These correspond to a static policy, with parallels to strong controllability in STNU literature. They would thus be *strongly* controllable with bounds on the probabilty of failure.

An example of a cc-pSTP is given as follows.

**Example 5.** *Consider the oceanography example introduced in Example 1, and suppose as in Example 2 that we require a scheduling policy such that the pSTN is controllable with probability of failure at most 5%. Figure 7 is a graphical depiction of the pSTN for this scenario. If we define a 5% chance-constraint over this pSTN then we obtain the following cc-pSTP, where* $E^c = \{A, B\}$, $E^u = \{C\}$, $C^r = \{\overrightarrow{AB}, \overrightarrow{BC}\}$, $C^u = \{\overrightarrow{AC}\}$, *and* $\Delta = 0.05$.

In real-world applications, we may also wish to optimize our policies based on defined performance metrics. For example, for an AUV mission, we may wish to minimize the total time of the mission. This motivates an optimal extension of the cc-pSTP.

**Problem 2** (Optimal cc-pSTP)**.** Given:

- $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$, *a pSTN; and*
- $\Delta \in [0, 1]$, *an upper bound on the risk of failure.*

Find:

- $\pi$, *a scheduling policy for* $E^c$ *based on outcomes of contingent constraint durations.*

Minimizing:

- $\mathbb{E}[c(\pi)]$, *the expected cost given the scheduling policy.*

Subject to:

- $\rho(\pi) \leq \Delta$, *the probability of policy risk being bounded by* $\Delta$.

$\mathcal{N}^p$ *is said to be* controllable with probability of failure at most $\Delta$ *if and only if such a policy* $\pi$ *exists.*

The optimal cc-pSTP adds an objective function to the problem formulation, calculated from the timing of the controllable events. However, in general policies assign times to controllable events based on the outcomes of contingent constraints. The timing of controllable events is thus probabilistic, and we minimize the objective function *in expectation.* The oceanography example may be extended with such an objective function as follows.

**Example 6.** *Consider the oceanography example outlined in Example 5. Suppose we wish to find the latest time at which we may arrive at the science site. Then, the objective to be minimized would be* $-t_B$.

In our subsequent treatment, we will focus on finding static scheduling policies – time tables for controllable events which do not depend on the outcomes of probabilistic contingent constraints. This particular variant of the problem is highlighted below.

**Problem 3** (Optimal static policy cc-pSTP)**.** Given:

- $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$, *a pSTN; and*
- $\Delta \in [0, 1]$, *an upper bound on the risk of failure.*

Find:

- $\pi : E^c \to \mathbb{R}$, *a full assignment for* $E^c$, *independent of the outcomes of contingent constraints.*

Minimizing:

- $c(\pi)$, *the cost of the static policy* $\pi$.

Subject to:

- $\rho(\pi) \leq \Delta$, *the probability of policy risk being bounded by* $\Delta$.

$\mathcal{N}^p$ *is said to be* strongly controllable with probability of failure at most $\Delta$ *if and only if such a policy* $\pi$ *exists.*

This problem is of practical interest. Oceanography missions are often drawn up with timetables for each day, as a common reference for large crews on vessels. In the use case of public transportation, we may solve for static policies which can then be published as timetables for departure times.

The focus on static policies also has implications on the formulation of the problem. In the general formulation in which we may adopt a dynamic policy, the timing of the controllable events may depend on the outcomes for the contingent constraints. As such, even in the presence of a policy, the actual timing of the events would be random variables, and we would need to evaluate the cost of the policy in expectation. In the static schedule setting, the objective function is over static assignments to controllable events, which are independent of the outcomes of the contingent constraints. The objective function is thus no longer evaluated over assignments which are probabilistic. This means that we can directly minimize the objective rather than minimizing the objective in expectation.

In the remainder of this article we present two efficient methods for solving pSTNs with static policies, and characterize their performance both analytically and empirically. Note that these two pSTN approaches preserve the algorithmic benefits of STNUs, by automatically reformulating pSTNs to STNUs using the concept of risk allocation (Ono and Williams, 2008), and then applying STNU reductions and consistency procedures to check pSTN strong controllability. In addition, both approaches offer automated procedures for deriving STNUs from pSTNs and for objectively evaluating the accuracy of the resulting STNUs with respect to the pSTN distributions and chance constraints.

## 4. Picard: Solving pSTNs by Risk Allocation and Convex Optimization

A cc-pSTP specifies a constraint on the probability of violating any one of a number of temporal constraints, which is an instance of the classic concept of a *joint chance-constraint* (Miller and Wagner, 1965). Evaluating whether a joint chance-constraint is satisfied is non-trivial, even for a given policy, except under strict assumptions. This motivates a body of literature on approximations within the stochastic optimization community. Prior approaches have included scenario- or particle-based evaluations (Nemirovski and Shapiro, 2006; Blackmore and Williams, 2007), which are numerically difficult to evaluate for smaller risk bounds. As an alternative, prior work has employed convex approximations (Nemirovski and Shapiro, 2007; Blackmore et al., 2010). Following this second approach, our main focus in this section is a solver called Picard, which draws on prior insights using the union bound to formulate a tractable convex approximation of the cc-pSTP. This approximation leverages the STNU reductions reviewed earlier. Picard encodes this approximation as a convex program, which it solves using an off-the-shelf commercial solver. This section thus expands on the treatment of the cc-pSTP given in Fang et al. (2014). In the next section, we will introduce an alternative solver called Rubato. Rubato builds upon the same convex approximation, but introduces a hybrid algorithm, which fully exploits STNU strong controllability and uses conflict-directed search to coordinate between subsolvers. Rubato's hybrid algorithm demonstrates significant improved performance over Picard's use of an off-the-shelf solver.

### 4.1 Approximate cc-pSTP

Consider our cc-pSTP encoding, which is based on prior approaches to encoding chance-constrained problems using the union bound. These approaches search for interval bounds on the uncertain variables whose cumulative distribution respects the chance constraint. To do so, they introduce decision variables that denote upper- and lower-bounds on each uncertain variables. In addition, they impose a constraint on the probability covered within the bounds. We take a similar approach for cc-pSTP. We introduce decision variables that bound the outcome of each uncertain duration. In addition, we introduce a constraint on the probability covered, which by the union bound is sufficient to satisfy the chance constraint (Problem 4).

**Problem 4** (Union Bound Approximation of cc-pSTP)**.** Given a cc-pSTP, comprised of:

- *pSTN* $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$*, and*
- $\Delta \in [0, 1]$*, an upper bound on the risk of failure,*

Find:

- *Lower-bounds* $L^p = \{l_c\}_{c \in C^p}$ *for contingent constraints* $C^p$*,*
- *Upper-bounds* $U^p = \{u_c\}_{c \in C^p}$ *for contingent constraints* $C^p$*, and*
- *static policy* $\pi'$*, for* $E^c$ *as a function of uncontrolled durations, mentioned in contingent constraints* $C^c = \{y - x \in [l_c, u_c]\}_{c \in C^p}$*,*

Minimizing:

- $c(\pi')$, *the cost of the static policy* $\pi'$.

Subject to:

- $$\sum_{c \in C^p} F_c(l_c) + \sum_{c \in C^p} 1 - F_c(u_c) \leq \Delta,$$

  *where each* $F_c$ *is the cumulative distribution for the duration of contingent constraint* $c \in C^p$.

In the original cc-pSTP, computing the risk of a candidate schedule is difficult because it involves integrating over the outcomes of a joint distribution that satisfies the requirements. In the approximate cc-pSTP we deal with this difficulty using the union bound, which states that

$$P(A \vee B) \leq P(A) + P(B)$$

for probabilistic events $A$ and $B$, regardless of the dependence between $A$ and $B$. We use this bound to factor the integration over the joint distribution to integrals over the individual random variables.

By solving the approximate cc-pSTP, we are trying to find a controllable STNU. We do so by finding interval bounds for the durations of contingent constraints that cover a large amount of probability mass. Intuitively, we guarantee the probability of scheduling success by accounting for a likely range of outcomes.

Problem 4 is thus the mathematical programming formulation of the problem we are looking to solve in this and subsequent chapters. While we present three approaches in the remainder of this paper (a monolithic encoding of the approximate problem as a numerical program; an iterative decomposition of the approximate problem; and a combination of the two approaches), all three are methods for solving the same problem.

We note that the Union Bound is an *upper bound* on the probability of failure, thus some conservatism is introduced. Notably, this formulation does not require the assumption of independence between durations in our prior work Wang and Williams (2015). We examine the correctness of the approximation in the next subection, and show that it is sound but not complete.

### 4.2 Soundness and Incompleteness

We prove that a policy $\pi$ obtained from solving the approximate cc-pSTP is also a solution to the exact cc-pSTP. The intuition behind the approach is as follows:

1. We restrict the probabilistic contingent constraints to bounded intervals, making them interval-bounded contingent constraints in a new STNU;

2. Suppose the STNU strongly controllable and $\pi$ is a valid schedule;

3. Suppose further that the subset has measure greater than $1 - \Delta$;

*Then* the schedule is consistent with probability at least $1 - \Delta$, hence the probability of failure is less than $\Delta$.

**Theorem 1.** *Suppose the policy $\pi$ and set of lower and upper bounds for contingent constraints $LU$ are the solutions to an approximate cc-pSTP with $\mathcal{N}^p = (E^c, E^u, C^r, C^p)$ and chance constraint $\Delta \in [0, 1]$, as given in Problem 4. Then $\pi$ is a solution to the corresponding exact cc-pSTP as given in Problem 1.*

*Proof.* Consider $\Delta$ the upper bound on the probability of failure for the requirement constraints $C^r$. Let $\Omega_\pi$ be the set of outcomes for which at least one requirement constraint is violated, as defined in Definition 7. Further, let $\Omega_{LU}$ the set of samples which give random durations outside intervals specified by $LU$.

Consider now any non-satisficing outcome $\omega \in \Omega_\pi$. Since $\pi$ and the lower and upper bounds $LU$ were the solutions for the approximate problem, we have $\pi$ valid with respect to all outcomes of the contingent constraints in $LU$. Therefore $\omega$ is not one of the outcomes in $LU$, so $\omega \in \Omega_{LU}$.

As this is true for any $\omega$ thus defined,

$$\Omega_\pi \subseteq \Omega_{LU}$$

Further, since $LU$ is a solution to the problem in Problem 4, we have

$$\Delta \geq P(\Omega_{LU}) \geq P(\Omega_\pi)$$

Noting that $P(\Omega_\pi)$ is exactly $\rho(\pi)$ the risk of schedule $\pi$, we thus have $\rho(\pi) < \Delta$ as required by the original problem.

□

While we have shown that the approximate cc-pSTP is sound, it is also incomplete. A solution may exist for the exact cc-pSTP but not for the approximation. Note that in general we have introduced conservatism, as $P(\Omega_{LU}) \geq P(\Omega_\pi)$. The actual risk of the returned schedule is less than the probability of contingent constraint values falling outside the returned intervals.

The approximation is thus an incomplete solution, in that although a solution $\pi$ may exist with $P(\Omega_\pi) \leq \Delta$, $P(\Omega_{LU}) \leq \Delta$ may not be true. That is, a schedule which is consistent with the required probability may exist, but there may not be a way to bound the intervals to cover enough probability mass. One such example is given as follows.

**Example 7** (Incompleteness of approximate cc-pSTP)**.** *Suppose we have the pSTN in Figure 8. By inspection, the timing of the* `start` *event does not impact the probability of failure.*

*Suppose we assign* `start` $= 0$*, so that the timing of the uncontrollable events take on the values of the contingent constraints. Figure 9 shows the feasible range of times for the uncontrollable events which result in temporal consistency.*

*However, the approximate cc-pSTP requires that we find a risk allocation which would carve out a rectilinear subset of the feasible region. Given any chance constraint, we can construct distributions so that the feasible region covers the required probability mass, but no such rectilinear subset exists.*

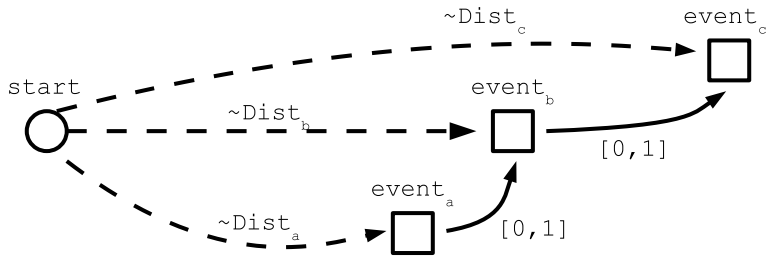


Figure 8: An example demonstrating the challenge involved in evaluating the risk of failure. For any schedule, we can characterize the subset of the sample space for which the schedule is inconsistent, but evaluating the measure of the subset is non-trivial.

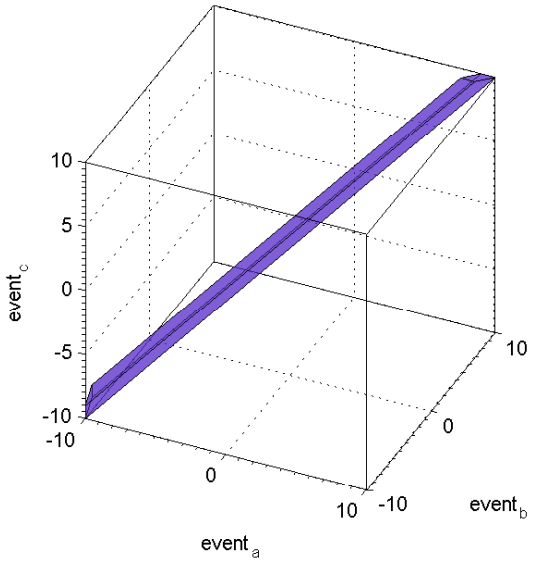


Figure 9: The plot shows part of the region in the space of the uncontrollable events which results in temporal consistency.

Although the approximation is incomplete, it allows us to evaluate the risk of candidate policies without performing multivariate integrations. We have thus traded off completeness for tractability, while retaining the soundness of solutions.

We have thus defined a cc-pSTP and a tractable approximation, in general terms with regard to controllability. In our subsequent treatment of the problem, we focus on strong controllability for cc-pSTPs. In our next subsection, we provide a numerical encoding for finding a strongly controllable schedule for the approximate cc-pSTP.

### 4.3 Picard: Convex Program Encoding of cc-pSTP

Given the approximate cc-pSTP, we now provide a solution method for deriving chance-constrained strongly controllable schedules. Intuitively, we choose the *most probable* set of outcomes of the contingent constraints, and schedule activated time points such that the timing constraints are satisfied for any combination of outcomes in this restricted set. The key insight behind our approach is that *by distributing the allowable risk amongst the contingent constraints, we can set-bound the outcomes.* This allows us to convert a pSTN into a simple temporal network with uncertainty (STNU), a well studied structure. We then make use of strong controllability reductions for STNUs to reduce the STNU to an STN, allowing us to reframe a cc-pSTP as a convex constrained programming problem, solvable with standard optimisers.

#### 4.3.1 Approach

We outline and motivate our approach before developing the details of our algorithm. We begin by noting that a cc-pSTP is an instance of a stochastic programming problem. A family of methods have been developed that reformulate the stochastic problem to a deterministic problem, by converting the stochastic constraints into deterministic constraints.

For example, optimising controls for stochastic dynamical systems involves bounding the extent of deviations from the mean, either by distributing the risk evenly (Van Hessem and Bosgra, 2006) or by optimising the distribution of risk (Ono et al., 2013). The process results in bounds on the state of the system, leading to reformulations as deterministic constraint optimisation problems.

#### 4.3.2 Convex Program Encoding of the Chance Constraint

Recall that in a cc-pSTP, we have an upper bound for the probability of temporal inconsistency. We *distribute this allowable risk* over the set of contingent constraints of the pSTN. For each contingent constraint, we then consider an interval subset of its possible outcomes according to the risk allocation. This allows us to represent the uncertainty in each contingent constraint as bounded intervals, giving us an STNU to check for strong controllability.

A direct approach is a convex program encoding. In this encoding we define our decision variables to be upper and lower bounds for the contingent constraints, as well as timing assignments to controllable events. We assert correctness with respect to the chance constraint by requiring that the sum of the probability mass above the upper bounds and below the lower bounds is less than our allowed risk. We enforce strong controllability through a set of linear constraints between event variables and upper and lower bound variables for the contingent constraints.

The concept of *risk allocation* has been used to provide scalable conservative approximations to risk evaluation in path planning with safety guarantees under probabilistic uncertainty (Blackmore and Williams, 2007; Ono and Williams, 2008). In this work, we also approximate the risk of a policy, by applying risk allocation to portion and distribute risk amongst the lower and upper bounds of contingent constraints.

Consider Figure 10. We are given a 10% upper bound on the probability of failure. For $k$ contingent constraints, we divide up the risk into $2k$ portions corresponding to the lower

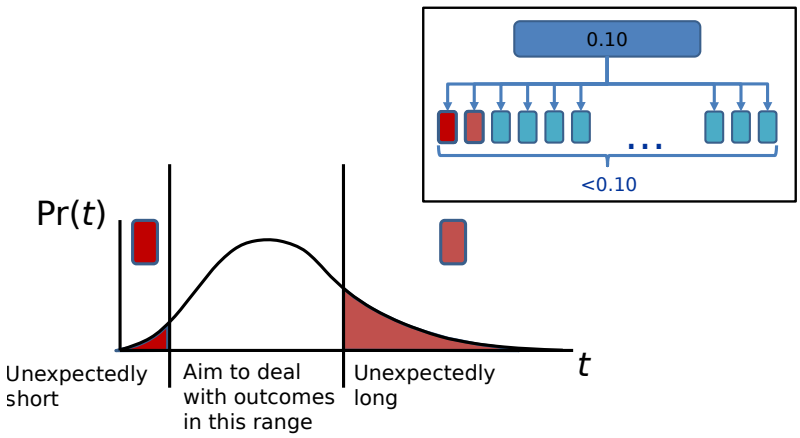


Figure 10: Converting from a probabilistic contingent constraint to a bounded contingent constraint via risk allocation.

and upper bounds for each contingent constraint, such that the sum of the portions is less than 10%. Let $\delta_l$ and $\delta_u$ be respectively the risk allocations to the lower and upper bounds of a contingent constraint. We may then find the lower and upper bounds as $F^{-1}(\delta_l)$ and $F^{-1}(1-\delta_u)$. Given this scheme for finding lower and upper bounds, the probability of any contingent constraint taking on a value outside the bounds is less than 10%.

This approach is sound in the sense that solution policies obtained will have risk of failure less than that allowed in the chance constraint. The inspiration behind risk allocation is the Union Bound, where $P(A \bigcup B) \leq P(A) + P(B)$ for events $A$ and $B$. We consider events in which the outcomes of contingent constraints land outside our prescribed bounds, intuitively being unexpectedly short or long in duration. The union bound thus guarantees that the probability of *any* contingent constraint in the network landing outside its bounds is at most the sum of the probabilities of each individual contingent constraint being unexpectedly short or long. This guarantees that the probability of observing an unaccounted for outcome is less than the risk specified in the chance constraint.

Note that the above procedure allows a convex program representation of a chance constraint. We proceed to discuss the encoding of simple temporal constraints.

#### 4.3.3 Convex Program Encoding of cc-pSTP

The previous sections have outlined that the chance constraint may be encoded as a nonlinear constraint, and that strong controllability may be encoded as a set of linear constraints. We now combine these two insights to produce a convex program that encodes the cc-pSTP. We begin by considering how this is done with our running example.

**Example 8.** *Recall Example 2, and assume that we would like to find a schedule with at least 95% probability of success. We only have one uncertain duration, $\overrightarrow{AC}$, and we introduce decision variables for the lower and upper bounds $l_{AC}$ and $u_{AC}$ respectively. We*

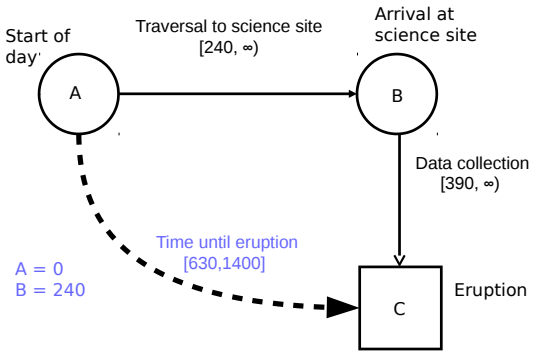


Figure 11: Example Picard solution to the oceanography scenario.

*further require* $P\left(\overrightarrow{AC} \notin [l_{AC}, u_{AC}]\right) \leq 0.05$ *in order to cover at least 95% of the possible outcomes.*

*In addition, we have controllable events,* $A$ *and* $B$*, so we introduce decision variables* $t_A$ *and* $t_B$ *to represent execution times of the events. We have linear constraint* $t_B - t_A \geq 240$ *as specified in the pSTN. Applying the strong controllability reductions, we also derive* $t_A + l_{AC} - t_B \geq 390$*. This is a convex, nonlinear encoding of the problem. Giving this encoding to a convex solver, for example SNOPT (Gill et al., 2005), we may obtain assignments:*

- $t_A = 0$, $t_B = 240$
- $l_{AC} = 630$, $u_{AC} = 1400$

*The solution pSTN and execution times are shown in Figure 11.*

More generally, we derive a system of constraints for a cc-pSTP as in Algorithm 1. For ease of reference, we denote the described convex encoding approach to solving cc-pSTNs as *Picard*. The first for-loop of Algorithm 1 adds two decision variables denoting lower and upper bounds for every contingent constraint, and notes the CDF $F_{d_{xy}}$ associated with each contingent constraint. The CDFs are used to calculate the probability mass lost by restricting the outcome of the contingent constraint. Note that both the CDF evaluating the probability mass discarded by the lower bound and the complement CDF evaluating that discarded by the upper bound are recorded.

In the second for-loop, the algorithm applies reductions to the contingent constraints in the pSTN. This corresponds to the constraints for the static schedule in Problem 4. Specifically, the set of reductions $\widehat{C}^{r-}$ are the linear constraints in the problem. The reductions are based on those for strong controllability. However, instead of fixed lower and upper bounds, the reductions performed allow the lower and upper bounds to be decided by the solver.

The reductions are similar to those proposed in Tsamardinos (2002). The key innovation is accounting for requirement constraints between two uncontrollable events, disallowed in previous work. This innovation follows from our reformulation of the pSTN. By representing

**Algorithm 1:** Convex program encoding for strong controllability of cc-pSTP

**input :** $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$ a pSTN
**output:** $B^p$ risk allocation bound variables on $C^p$,
$F$ chance constraint function, and
$\widehat{C}^{r-}$ reductions

$B^p \leftarrow \emptyset$, $\widehat{C}^{r-} \leftarrow \emptyset$,$F \leftarrow \emptyset$
**for** *each* $c_{XY} \in C^p$ **do**
  $B^p \leftarrow B^p \cup \{l_{XY}, u_{XY}\}$
  $F \leftarrow F \cup \{F_{c_{XY}}; 1 - F_{c_{XY}}\}$
**for** *each* $c_{XY} \in C^r$ **do**
  $a, b \leftarrow$ lower and upper bounds for $c_{XY}$ respectively
  **if** $X \in E^u \wedge Y \in E^u$ **then**
    Let $c_{t_iX}$, $c_{t_jY}$ be contingent constraints ending in $X$ and $Y$, $[l_X, u_X]$ and $[l_Y, u_Y]$ corresponding bounds on the contingent constraints
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{t_j - t_i + l_Y - u_X \geq a\}$
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{-t_j + t_i - u_Y + l_X \geq -b\}$
  **else if** $Y \in E^u$ **then**
    Let $c_{t_jY}$ be contingent constraint ending in $Y$, $[l_Y, u_Y]$ corresponding bounds on contingent constraints
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{t_j - X + l_Y \geq a\}$
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{-t_j + X - u_Y \geq -b\}$
  **else if** $X \in E^u$ **then**
    Let $c_{t_iX}$ be contingent constraint ending in $X$, $[l_X, u_X]$ corresponding bounds on contingent constraints
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{Y - t_i - u_X \geq a\}$
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{-Y + t_i + l_X \geq -b\}$
  **else**
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{Y - X \geq a\}$
    $\widehat{C}^{r-} \leftarrow \widehat{C}^{r-} \cup \{-Y + X \geq -b\}$

stochasticity with interval-bounded contingent durations, we have a natural mapping from the STNU structure, allowing us to transcribe Case 3 in a probabilistic context.

Applying reductions in Algorithm 1, we are thus able to transform the approximate problem given in Problem 4 to the following form.

**Problem 5.** Given a cc-pSTP, comprised of:

- *pSTN* $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$, *and*
- $\Delta \in [0, 1]$, *an upper bound on the risk of failure,*

*Solve the numerical program*

$$\begin{aligned} \min_{E^C, B^P} \quad & c(\mathbf{E^c}) \\ s.t. \quad & \sum_{i \in 2|E^u|} F_i(B_i^P) \leq \Delta \\ & C^{r-} \end{aligned}$$

*for* $B^p$, $F$, *and* $\widehat{C}^{r-}$ *from Algorithm 1.*

Note that the constraints are in a form solvable with off-the-shelf nonlinear solvers. The reductions comprise the majority of the constraints, and are linear in the decision variables $\pi$ and $LU$. The only possible source of nonlinearity is the chance-constraint.

We have thus provided a mathematical encoding for finding strongly controllable schedules to cc-pSTPs. However, we may leverage known algorithms from the STNU literature to solve problems involving pSTNs. While the monolithic encoding casts strong controllability as a mathematical programming problem, the STNU community has polynomial time techniques for checking strong controllability. In the next section, we show how to leverage these algorithms to separate out the problems of risk allocation and controllability checking, resulting in an iterative algorithm with conflict-guided risk allocation.

## 5. Hybrid Solver Using Conflict-Directed Consistency

The previous section established the notion of temporal risk allocation in solving the cc-pSTP. We showed that one way to find a chance-constrained schedule was to produce a risk allocation whose corresponding STNU was also strongly controllable. However, we essentially treated the temporal aspect of the problem as a linear program and fell back on generic mathematical programming.

By contrast, the family of work on STNs and their extensions exploits the insight that the temporal constraints are essentially difference constraints. This has allowed for specialized algorithms that center around negative cycle detection (Dechter et al., 1991; Morris and Muscettola, 2005). These are particularly efficient methods for detecting when a STN is inconsistent, or when an STNU is uncontrollable. This key insight has been used to discover how to relax temporal constraints given uncertainty over activity durations (Yu et al., 2017).

These insights motivate an alternative to a monolithic encoding of the cc-pSTP problem. Rather than specifying the complete list of temporal constraints and reductions at the start,

we may generate candidate risk allocations, and use efficient methods for identifying why the resulting STNU is not strongly controllable. These can be summarised as numerical constraints to be satisfied in subsequent risk allocations. Effectively, rather than solving one numerical program with a large number of constraints, we may solve a sequence of numerical programs with a smaller number of constraints. This section expands on this approach, first given in Wang and Williams (2015), with additional insight into edge cases for when the algorithm will and will not perform well.

The approximate cc-pSTP in Problem 4 may thus be decomposed into two portions: 1) the nonlinear programming portion containing the risk allocation; and 2) the temporal constraints portion. The risk allocation can be used to suggest possible STNUs, to be checked for controllability using specialized methods working on temporal networks.

### 5.1 Approach

The key feature of an STNU being strongly controllable is that it is equivalent to a consistent STN. In turn, recall from Section 2 that an STN is consistent if and only if its distance graph contains no negative cycles. Negative cycle detection algorithms only work on graphs with fully instantiated edge weights. In contrast, we are trying to *construct* an STNU without negative cycles by performing risk allocation to contingent constraints. Therefore, such algorithms do not apply directly to our *parameterized* STNU.

However, given a grounded STNU, where the lower and upper bounds on contingent constraints have been found via risk allocation, it is straightforward to verify whether it is controllable. We would simply construct the equivalent STN and check for negative cycles.

In the absence of negative cycles, we would derive a schedule for the STN.The schedule would be a solution to the cc-pSTP as long as the original risk allocation had satisfied the chance constraint. The presence of a negative cycle proves that the risk allocation resulted in a STNU that is not strongly controllable. A different risk allocation would be needed, in which the cycle in the resulting STNU is no longer negative.

This suggests an iterative approach where we use negative cycles to repeatedly constrain the risk allocation. In essence, we would be searching over the space of STNUs, using discovered negative cycles as conflicts to drive the search. This approach is codified in an algorithm called Rubato.

### 5.2 Rubato: Iterative Solution of cc-pSTP

Rubato splits the solution process into a master problem and a subproblem. The master problem handles the chance constraint by generating a risk allocation that respects the risk bound. The subproblem then verifies the corresponding STNU for strong controllability. Only when both parts are successful do we have a solution to the cc-pSTP.

Given a risk allocation, the resulting STNU may not be strongly controllable. The algorithm must then identify the negative cycle that violated strong controllability, and generate another risk allocation that ensures the cycle is nonnegative. Recall that the cycle lies in the distance graph of the STN that was derived from the STNU using strong controllability rules. Thus, we would need to: 1) map the edges of that cycle back to the edges of the STNU they were derived from; 2) *undo* the assignments to risk allocation

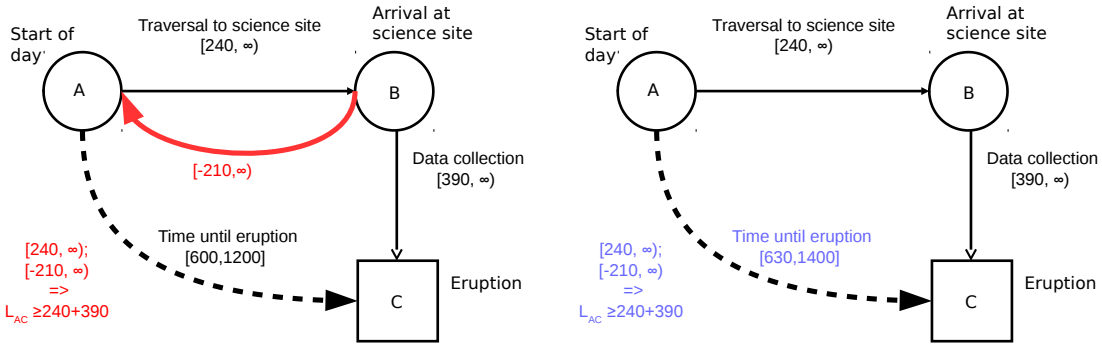


Figure 12: Example Rubato solution to the oceanography scenario.

variables in those edges; and 3) add up the weight expressions of all those edges. This results in a linear expression in terms of the risk allocation variables.

The next iteration of the risk allocation would thus try to make this linear expression nonnegative, while still respecting the risk bound. The negative cycle avoidance is encoded as a linear inequality to be satisfied for all subsequent risk allocations. Note that in the next round of risk allocation, we would be recalculating the assignments to all of the risk allocation variables, not just those involved in the newly discovered negative cycle. Intuitively, this is because the new linear constraint may require a greater sum of risk to be allocated to the contingent constraints involved in the negative cycle. If we fix the risk allocated to the other contingent constraints, we may find that we can not allocate the required additional amount of risk, and falsely conclude that the problem is infeasible.

This process would continue until one of two things happens: Either the subproblem returns no conflict, and so we have a solution, or the master gets too constrained to produce a risk allocation, in which case we declare no solution. The termination and completeness of this process are discussed in the next subsection. For now, we demonstrate it on the running example and present the pseudocode of Rubato.

**Example 9.** *Recall Example 2. We wish to find an execution strategy which will work with probability at least 95%. We thus first divide up the risk evenly between the lower and upper bounds of the uncertain duration* $\overrightarrow{AC}$*, resulting in the bounds* $[600, 1200]$*. We apply standard strong controllability checking, and find that the proposed STNU is not strongly controllable. Based on STNU theory, this is due to the negative cycle* $-240 - (-210) < 0$*. Note that the* $-240$ *comes from the lower bound of* $\overrightarrow{AB}$ *and* $-210$ *comes from the lower bound of* $\overrightarrow{BC}$ *subtracting* $L_{AC}$*, the lower bound of* $\overrightarrow{AC}$*. Rearranging the terms in the negative cycle, we find that any assignment to* $L_{AC}$ *must satisfy* $L_{AC} \geq 240 + 390$ *for the resulting STNU to be strongly controllable.*

*The linear constraint on* $L_{AC}$ *is then inserted when we propose the next candidate value for the lower bound. This results in bounds* $[630, 1400]$ *for uncertain duration* $\overrightarrow{AC}$*. The new STNU is then checked for strong controllability. It is shown to be strongly controllable, and therefore any execution strategy for the STNU also satisfies the original cc-pSTP.*

**Algorithm 2:** Rubato - Feasibility

**input :** $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$ a pSTN, and
$\Delta \in [0, 1]$ a risk bound
**output:** $\pi : \mathbb{R}^{|C^c|} \to \mathbb{R}^{|E^c|}$, a constant scheduling policy to $E^c$, and
$LU$ risk allocation bounds on $C^p$

1 $B^p \leftarrow \{l_i, u_i | c_i \in C^p\}$
2 constraint list $\leftarrow \left\{ \left[ \sum_{c \in C^p} F_c(l_c) + 1 - F_c(u_c) \leq \Delta \right] \right\}$
3 $\widetilde{\mathcal{N}}^u \leftarrow$ `MapToParametricSTNU(`$\mathcal{N}^p$, $B^p$`)`
4 **while** *True* **do**
5 ok, $LU \leftarrow$ `Solve(`constraint list`)`
6 **if** *not* ok **then return** no solution
7 $\mathcal{N}^u$, edge mapping $\leftarrow$ `Instantiate(`$\widetilde{\mathcal{N}}^u$, $LU$`)`
8 ok, $\pi$, cycle = `CheckSC` ($\mathcal{N}^u$)
9 **if** ok **then return** $\pi$, $LU$
10 $\widetilde{\text{cycle}} \leftarrow$ `Uninstantiate(`cycle, edge mapping`)`
11 `Append(`constraint list, $\left[\texttt{Weight(}\widetilde{\text{cycle}}\texttt{)} \geq 0\right]$`)`

Rubato, given in Algorithm 2, begins by defining the initial master problem. The variables of the problem are lower and upper bounds on each probabilistic duration of the pSTN, representing the risk allocation to be found. The risk allocation master is initiated with only the chance constraint. The risk allocation variables, applied to the pSTN, establish the concept of a parametric STNU $\widetilde{\mathcal{N}}^u$, where the contingent constraints are bounded, but their bounds are uninitialized.

The risk allocation master, comprising the chance constraint initially added to the constraint list, as well as the subsequent linear constraints derived from the negative cycles, is solved using a nonlinear program, much like Picard. If the solver finds a solution, then the master can project the assignment onto the parametric STNU to instantiate a grounded STNU. The grounded STNU uses the values for the lower and upper bounds of contingent constraints found via risk allocation. A mapping of edges from the parametric STNU to the grounded one is also recorded, for later use if conflicts will need to be resolved.

If the solver could not find a solution at this point, then Rubato would return without a valid policy. Alternatively, if a valid risk allocation is found, the decision variables provide the lower and upper bounds for contingent constraints in an STNU.

Leveraging STNU controllability theory, we check whether the instantiated STNU is strongly controllable. If so, then the function can also return a schedule for that STNU. When such a schedule is produced, we declare it a solution to the cc-pSTP, and return. Otherwise, a negative cycle must be presented as evidence of the STNU's uncontrollability.

All future risk allocations must avoid this negative cycle. First, the cycle is mapped back to its analog in the STNU. We do so by performing reverse lookups on the parametric-to-grounded edge mapping that was created alongside the grounded STNU. Then we collect the weight expressions along that analog cycle, constants and variables alike, and state that the

expression for their sum must be nonnegative. That statement, inserted into the constraint list, represents the learning by the master of temporal conflicts from the subproblem.

We provide a run-through of how Rubato would work on the running example in Figure 12 in the following Example.

**Example 10.** *First, the variables and constraints of the master problem would be initialized as follows:*

$$\textit{risk allocation variables} = \{l_1, u_1\}$$
$$\textit{constraint list} = \{[F_1(l_1) + 1 - F_1(u_1) \leq 0.05]\}$$

*The function $F_1$ represents the cumulative distribution of the probabilistic duration for time until eruption.*

*The master passes the list of variables and constraint to a nonlinear solver and asks for a solution. In return, the solver, whose details we do not really care about, could yield the following solution: Assign 600 to the lower bound variable $l_1$, and assign 1200 to the upper bound variable $u_1$. We can verify by inspection that the solution satisfies the chance constraint, since the bounds lie at the $\pm 2\sigma$ points of the distribution.*

*When this STNU is passed to the subproblem, it will be determined that it is not strongly controllable, so we do not get a policy. However, we do get a negative cycle consisting of: the lower bound of AC, the negated lower bound of BC, and the negated lower bound of AB. Indeed, we can verify that*

$$l_{AC} - l_{BC} - l_{AB} = 600 - 390 - 240 = -30 < 0$$

*We map this negative cycle back into the parametric STNU. That means the condition we insert back in the master's constraint list is:*

$$l_{AC} - l_{BC} - l_{AB} \geq 0$$

*Now we begin the second iteration, and the master's job is to amend the original risk allocation to satisfy the new cycle constraint, while preserving the chance constraint. When we send this request through the nonlinear solver, it could assign 630 to $l_1$ and 1400 to $u_1$.*

*The updated grounded STNU is also shown in Figure 12. We now no longer account for outcomes in which the eruption occurs less than 630 minutes from the start of day, but allow the eruption to occur up to 1400 minutes after the start of day. We can also verify this risk allocation satisfies the chance constraint.*

$$F_1(630) + 1 - F_1(1400) = 0.0359 + 1 - 0.9996 < 0.05$$

*When we pass this STNU to the subproblem, we will find there are no negative cycles. Therefore, we will get a strongly controllable scheduling policy for the STNU, which decides when the controllable timepoints are executed. In our case, there are only two controllable timepoints: the start event and the beginning of data collection. A strongly controllable policy would say that we have to start collecting data at 240 minutes from start of day. That is because, according to our risk allocation, we have to allow the eruption to happen as early as 630, and we need at least 390 minutes to perform data collection. That means we can take up to 240 minutes to traverse to the site – we have to travel at our maximum speed.*

### 5.3 Termination

Rubato, like the monolithic numerical encoding, solves the approximate cc-pSTP, and thus inherits the soundness and completeness properties with respect to the original cc-pSTP. However, Rubato is an iterative algorithm, explictly incorporating techniques from the STNU literature. In this subsection, we discuss the termination conditions of Rubato.

**Theorem 2** (Rubato Termination)**.** *Rubato, described in Algorithm 2, terminates in finite time, given a cc-pSTP with a finite number of requirement constraints.*

*Proof.* Note first that the subprocedures in the loop are finite time. We assume that the nonlinear solver terminates given any input. Further, strong controllability checking is done in polynomial time (Vidal and Fargier, 1999).

We may thus prove finite time termination by showing that Rubato only runs for a finite number of loops. For a complete loop through Lines 4 to 11, there must have been a solution found by the nonlinear solver (Line 6), and the resulting STNU must not be strongly controllable (Line 9). If either condition is not met, the algorithm terminates via the respective return statements.

Note that, for each completed loop, one new negative cycle must be found by construction - the risk allocation performed by the nonlinear program must explicitly resolve all previous negative cycles, and the resulting STNU is not strongly controllable. The argument thus hinges on showing that there is only a finite number of possible negative cycles.

Recall that the underlying pSTN from the cc-pSTP is first mapped to a STNU, which is then reduced to a STN for strong controllability checking. The negative cycles are essentially combinations of the reduced requirement constraints in the resulting STN distance graph. Each requirement constraint adds two edges in the distance graph, each of which may feature up to once in a negative cycle. Thus, for $\mathcal{T}$ requirement constraints, there are at most $2^{2\mathcal{T}}$ possible negative cycles.

The number of negative cycles is thus finite for a cc-pSTP with finite requirement constraints. There is thus a finite number of times Rubato may loop, and Rubato thus returns in finite time.

□

So Rubato terminates because it cannot go on forever collecting cycles. That does not mean, however, that there might not be a very large number of cycles in the STNU, waiting to be discovered.

Contrary to prior analysis (Wang and Williams, 2015), the number of negative cycles is not bounded by the number of linear constraints in the monolithic encoding given in Picard. Again, the negative cycles are combinations of the linear constraints in the monolithic encoding. Each linear constraint in Picard may feature in multiple negative cycles discovered by Rubato

Even so, a cycle typically combines multiple temporal constraints into one condition, and not all cycles may need to be discovered. In practice, we observe that the number of negative cycles detected is usually fewer than the number of requirement constraints. Rubato thus typically solves sequences of nonlinear programs with much fewer constraints, and thus is still efficient with respect to the total computation time.

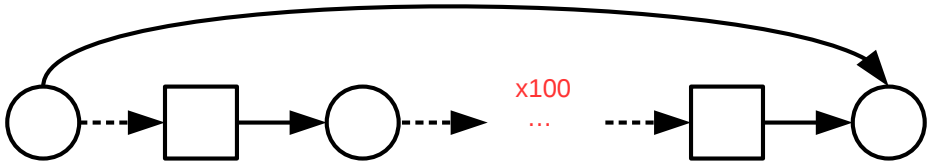


Figure 13: A pSTN of probabilistic constraints in series.

### 5.4 Efficiency Relative to Picard

At the heart of the cc-pSTP is the tension between the chance constraint and the temporal constraints. The chance constraint wants to push the uncontrollable bounds outwards to reduce the risk of failure, while the temporal constraints wants to push them inwards to make the STNU more controllable. To resolve this tension, both Rubato and Picard let the solver handle the nonlinear gradient analysis needed to move the risk allocation around.

Rather than using a general purpose numeric program, the approach taken in Rubato separates out the task of verifying temporal controllability into a subproblem, which is then handled by an efficient dedicated method. In this subsection, we present two pathological examples to illustrate the conditions under which the decomposition performs well, and when the decomposition performs badly.

Consider the pSTN in Figure 13. It consists of a sequence of 100 probabilistic durations, each followed by an infinite controllable wait. The entire sequence is constrained by an overall deadline. Rubato would require up to two iterations to process this network against a chance constraint. The first iteration, as always, is to find a risk allocation that satisfies only the chance constraint. In the event that the uncontrollable upper bounds add up to more than the deadline, the subproblem will find this conflict and add that cycle into the second iteration of the master. This cycle summarizes the only real temporal condition; the uncontrollable lower bounds can never be involved in a cycle, because the strong controllability rules attach them to the infinities on their respective controllable waits, which essentially means no constraint. So the final master problem to be solved by Rubato would contain only two constraints.

In contrast, Picard would create 101 additional variables for the controllable events, and deduce the 100 controllable lower bound constraints between these events (the 100 controllable upper bound constraints all become infinity and are thus ignored, like in Rubato). Adding those on top of the 200 lower and upper bound variables, the deadline's lower and upper bound constraint, and the chance constraint, that's a total of 301 variables and 103 constraints sent to the solver. The main cost of the solver will be to compute first- and second-order partial derivatives for all constraints with respect to all variables. On the other hand, even though Rubato requires two iterations, it does away with the 101 event variables and has only two constraints to compute derivatives for. Rubato thus allows a more compact encoding of the problem.

Here is a negative example. Figure 14 depicts another pSTN. This time all the activities are in parallel, so there are only two controllable events—the start and end—and the waits after each probabilistic duration are 30 minutes instead of infinite. Assume that each

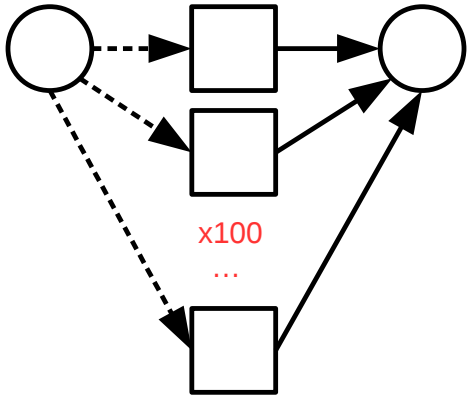


Figure 14: A pSTN with probabilistic constraints in parallel.

probabilistic duration is normally distributed and followed by a $[0, 30]$ constraint as follows:

$$\begin{aligned} N(85, 1), [0, 30] \\ N(84.9, 1), [0, 30] \\ N(84.8, 1), [0, 30] \\ \dots \\ N(75.1, 1), [0, 30] \end{aligned}$$

Applying strong controllability rules will effectively convert each horizontal "thread" into a controllable duration between the start and end, with lower bound $u_i$ and upper bound $l_i + 30$. Assuming a 99% chance constraint, a plausible initial risk allocation in the first round of Rubato would be as follows.

$$\begin{aligned} \{l_1, u_1\} &= \{70, 100\} \\ \{l_2, u_2\} &= \{69.9, 99.9\} \\ \{l_3, u_3\} &= \{69.8, 99.8\} \\ &\dots \\ \{l_{100}, u_{100}\} &= \{60.1, 90.1\} \end{aligned}$$

These bounds are at the $\pm 15\sigma$ points of each distribution, so there is practically no risk of violating the chance constraint. Now say we discover in the subproblem that $l_2 + 30$ is less than $u_1$. This forms a negative cycle, so we would send this back to the master, and it could be resolved by raising $l_2$ from 69.9 to 70. In the same manner, we would discover that every $l_{i+1} + 30$ is less than $u_i$, so we would have a total of 99 corrections to make, raising

each $l_{i+1}$ by 0.01.

$$\begin{aligned}\{l_1, u_1\} &= \{70, 100\}\\ \{l_2, u_2\} &= \{70, 99.9\}\\ \{l_3, u_3\} &= \{69.9, 99.8\}\\ \{l_4, u_4\} &= \{69.8, 99.7\}\\ &\dots\\ \{l_{100}, u_{100}\} &= \{60.2, 90.1\}\end{aligned}$$

However, now we will find $l_3 + 30$ is less than $u_1$, and that in general every $l_{i+2} + 30$ is less than $u_i$ by 0.01. So that is another 98 rounds of corrections. Continuing in this fashion, there will be $99 + 98 + ... + 1 = 4950$ master-subproblem iterations before we arrive at the following risk allocation.

$$\begin{aligned}\{l_1, u_1\} &= \{70, 100\}\\ \{l_2, u_2\} &= \{70, 99.9\}\\ \{l_3, u_3\} &= \{70, 99.8\}\\ \{l_4, u_4\} &= \{70, 99.7\}\\ &\dots\\ \{l_{100}, u_{100}\} &= \{70, 90.1\}\end{aligned}$$

Note that these bounds, placed on their respective distributions, all lie beyond $\pm 5\sigma$, and the cumulative probability at $5\sigma$ is much less than 1/100th of a percent, so the chance constraint is still satisfied. Therefore, this is a solution to 200 variables that could have taken Rubato – including the final round for verification – 4951 rounds to find, accumulating 4950 conflicts along the way.

Now let's see how Picard would have handled the problem. In addition to the risk allocation variables, it is also solving for the start and end event times, so there are a total of 202 variables. Then Picard will apply strong controllability to each thread to directly encode a controllable duration of $[u_i, l_i + 30]$ from the start to the end. That's merely 200 linear constraints plus a chance constraint, which will be much easier to solve than Rubato's accumulated list of 4951 constraints. The key reason for this difference is that the parallel structure turns every pair of constraints into a cycle, so the number of cycles is quadratic in terms of the number of constraints, and Rubato may have to find them all.

It is worth noting that Picard's encoding is close to optimal. We can intuitively understand the constraints of the problem as: every $u_i$ has to be less than or equal to every $l_i + 30$. Rubato interprets this literally as a quadratic number of constraints, while Picard attaches every constraint to the same start and end variables. If we notice that the expression (end − start) appears in every constraint in Picard, we can replace that with a new variable $m$ such that $u_i \leq m$ and $m \leq l_i + 30$, which would be the most efficient encoding.

These examples are extreme cases, but they illustrate the mechanisms that affect the performance of Rubato. In the second example, we constructed the risk allocation and chose the conflicts in an order that would guarantee the discovery of every cycle. So we could have been unlucky, and a better conflict discovery order might have helped. However, specialized

conflict extraction algorithms are outside the scope of this paper, and we leave it to the experiments section to see how a generic one performs on various scenarios. Together, the above examples and benchmarks should give some insight on when to use one approach over the other.

### 5.5 Rubato for Optimization

The preceding discussion, as well as our prior work on the iterative approach (Wang and Williams, 2015), has concentrated on finding feasible solutions to cc-pSTPs, without reference to the objective function. In this paper, we extend these results by considering how optimization may be performed.

A straight forward method to optimization, while leveraging negative cycle detection, is to incorporate the objective function directly into the risk allocation step of Rubato, as in Algorithm 3.

Note that the cost function is defined over the static policy, and hence the assignments to the controllable events. Thus, we can not perform risk allocation only over the lower- and upper-bounds of the contingent constraints. Instead, we must perform risk allocation and optimize over assignments to the controllable events at the same time, as in Line 5.

The resulting assignments to the controllable events are then reflected in controllability checking, by augmenting the resulting STNU. For every assignment $a$ to controllable event $e$, a requirement constraint $[a, a]$ is added from a reference controllable event to $e$ - intuitively, this sets the timing of the controllable event. The remainder of the algorithm is the same as that given in Algorithm 2.

**Algorithm 3:** Rubato - Optimizing during Risk Allocation

**input :** $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$ a pSTN,
$\Delta \in [0, 1]$ a risk bound, and
$c$ an objective defined over the static policy
**output:** $\pi : \mathbb{R}^{|C^c|} \to \mathbb{R}^{|E^c|}$, a constant scheduling policy to $E^c$, and
$LU$ risk allocation bounds on $C^p$

1 $B^p \leftarrow \{l_i, u_i | c_i \in C^p\}$
2 constraint list $\leftarrow \left\{ \left[ \sum_{c \in C^p} F_c(l_c) + 1 - F_c(u_c) \leq \Delta \right] \right\}$
3 $\widetilde{\mathcal{N}}^u \leftarrow$ `MapToParametricSTNU(`$\mathcal{N}^p$, $B^p$`)`
4 **while** *True* **do**
5 ok, $LU$, $\pi \leftarrow$ `Optimize(`$c$, constraint list`)`
6 **if** *not* ok **then return** no solution
7 $\mathcal{N}^u$, edge mapping $\leftarrow$ `Instantiate(`$\widetilde{\mathcal{N}}^u$, $LU$, $\pi$`)`
8 ok, $\pi$, cycle = `CheckSC` ($\mathcal{N}^u$)
9 **if** ok **then return** $\pi$, $LU$
10 $\widetilde{\text{cycle}} \leftarrow$ `Uninstantiate(`cycle, edge mapping`)`
11 `Append(`constraint list, $\left[$`Weight(`$\widetilde{\text{cycle}}$`)` $\geq 0 \right]$`)`

However, this straightforward approach was observed empirically to be inefficient. This can be attributed to the inclusion of the objective function in the risk allocation step, in particular the need to make explicit assignments to the controllable events.

Consider first the risk allocation step. With the inclusion of the controllable events, the nonlinear program is now defined over a larger number of variables. Each risk allocation thus takes longer to compute.

In addition, the results from risk allocation are more likely to be infeasible. Recall that we introduce additional requirement constraints reflecting the assignments to the controllable events. These additional requirement constraint remove any flexibility in the STNU - the controllable events are effectively already scheduled by that point. Even if the bounds on the contingent constraints would have allowed us to produce a strongly controllable STNU, the strong controllability check would still fail if the static schedule was not consistent for all possible outcomes for the contingent constraints. This results in a higher number of risk allocations. As an alternative, we perform Rubato without the objective until a first feasible solution is found. In this version of the optimizing Rubato algorithm, we only perform risk allocation with the objective function after a first chance-constrained solution is found. In this way, we quickly discover a number of negative cycles in the process of finding a consistent solution, which are then used when trying to find an optimal solution. This approach is outlined in Algorithm 4.

## 6. Numerical Results

The algorithms proposed were evaluated on a set of 138 benchmark problems, inspired by autonomous underwater vehicle (AUV) scenarios. In each of the scenarios, a number of AUVs must coordinate to explore a series of promising locations. Each vehicle must perform a number of dives, and for each dive must perform a number of surveys.

The promising locations were randomly generated from the region within 10km of (33.251, -121.555), in the North Pacific. The vehicle traversal durations are modeled as normally distributed random variables with parameters derived from distance traveled and an average vehicle speed uniformly sampled between 10km/h and 20km/h, and each vehicle must spend a minimum amount of time exploring each area. The benchmark set contained 900 randomly generated scenarios. For each scenario, there were between 1 and 12 robots, up to 5 dives for each robot, and up to 4 activities per dive.

For each scenario, we required chance-constrained schedules with risk bounds of 5%, 10%, 20% and 40% respectively. We used the makespan of the network as the objective function. For brevity, we present only a summary of the empirical results in this section. The full tables of results are given in the supplemental material.

We compare the performances of the following algorithms:

1. Rubato: the iterative consistency checking algorithm given in Wang and Williams (2015). This implementation does not consider the objective function, and only tries to find a feasible solution.

2. Picard: the monolithic encoding as given in Fang et al. (2014) and Section 4. This encoding does consider the objective function, and tries to find the optimal solution

**Algorithm 4:** Rubato - Feasible Seed

**input :** $\mathcal{N}^p = \langle E^c, E^u, C^r, C^p \rangle$ a pSTN,
$\Delta \in [0, 1]$ a risk bound, and
$c$ an objective defined over the static policy

**output:** $\pi : \mathbb{R}^{|C^c|} \to \mathbb{R}^{|E^c|}$, a constant scheduling policy to $E^c$, and
$LU$ risk allocation bounds on $C^p$

$B^p \leftarrow \{l_i, u_i | c_i \in C^p\}$

constraint list $\leftarrow \left\{ \left[ \sum_{c \in C^p} F_c(l_c) + 1 - F_c(u_c) \leq \Delta \right] \right\}$

$\widetilde{\mathcal{N}}^u \leftarrow$ `MapToParametricSTNU(`$\mathcal{N}^p$, $B^p$`)`

$c' \leftarrow 0$

feas $\leftarrow$ False

**while** *True* **do**

  ok, $LU$, $\pi \leftarrow$ `Optimize(`$c'$, constraint list`)`

  **if** *not* ok **then return** no solution

  $\mathcal{N}^u$, edge mapping $\leftarrow$ `Instantiate(`$\widetilde{\mathcal{N}}^u$, $LU$, $\pi$`)`

  ok, $\pi$, cycle = `CheckSC` ($\mathcal{N}^u$)

  **if** ok **then**

    **if** feas **then return** $\pi$, $LU$

    **else**

      feas $\leftarrow$ True

      $c' \leftarrow c$

  $\widetilde{\text{cycle}} \leftarrow$ `Uninstantiate(`cycle, edge mapping`)`

  `Append(`constraint list, $\left[\texttt{Weight(}\widetilde{\text{cycle}}\texttt{)} \geq 0\right]$`)`

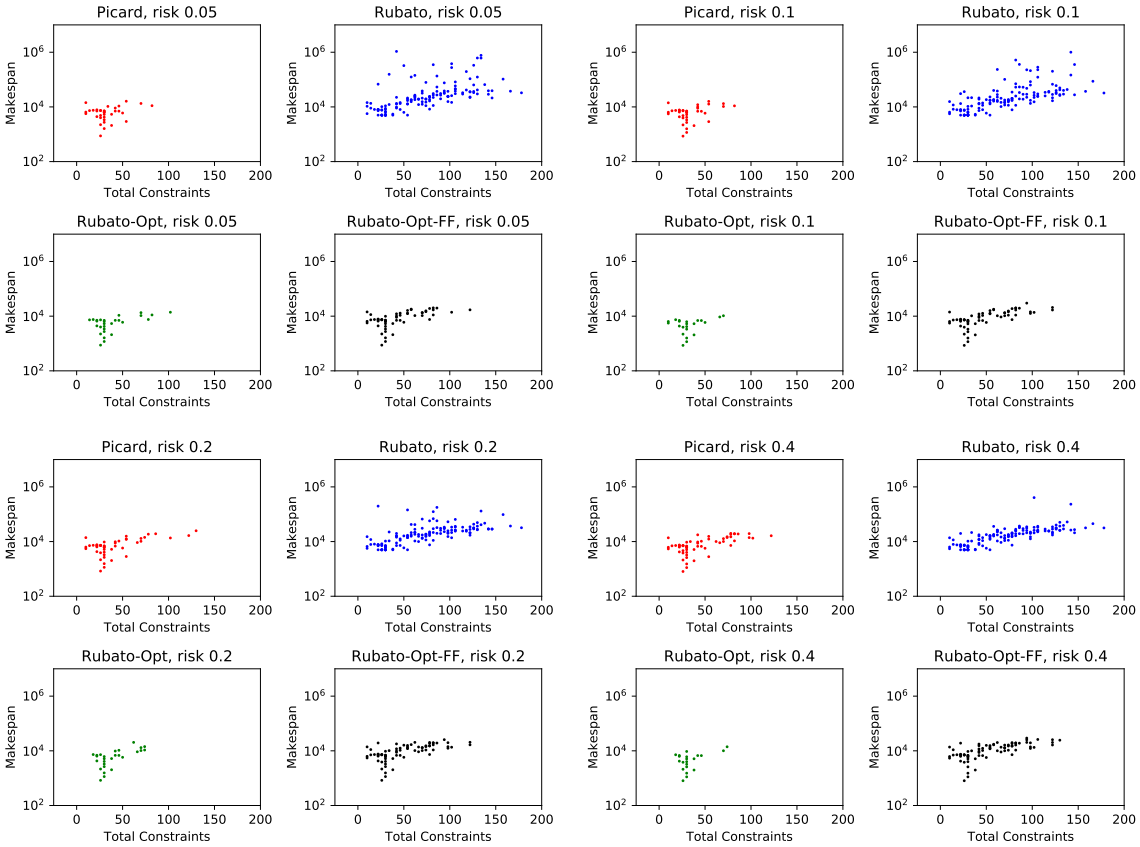


Figure 15: Quality of solution for benchmark scenarios.

without decomposing the problem and exploiting efficient methods for the controllability subproblem.

3. Rubato-Opt: the optimal version of Rubato, in which the objective function is used when performing each iteration of risk allocation, as presented in Section 5.

4. Rubato-Opt-FF: the feasible-first version of Rubato-Opt, in which a feasible solution to the chance-constrained problem is found using Rubato, before the objective is added to the nonlinear program for risk allocation, as presented in Section 5.

As Rubato exploits efficient algorithms for the strong controllability subproblem, and stops at the first feasible solution, we expect the algorithm to be fast. However, this comes at the cost of suboptimal solutions. We provide evidence that the suboptimality can be significant relative to Picard. This motivates the search for an algorithm which optimizes with respect to an objective function, while maintaining a runtime performance similar to Rubato. We test two such approaches, Rubato-Opt and Rubato-Opt-FF, both proposed in Section 5. Our results show that while Rubato-Opt is faster than Picard, it is still significantly slower than both Rubato and Rubato-Opt-FF.

Figure 15 shows the makespans of the networks obtained using each solution method, compared against the total number of constraints. The total number of constraints can be obtained as the number of simple temporal constraints in the problem, with one more representing the nonlinear chance constraint.

The networks derived with the consistency algorithm typically result in makespans which are far higher than the optimal, motivating the use of optimisation methods. However, we may also note that the optimising algorithms solved far fewer instances. This is a direct consequence of the difficulty of finding an optimal solution relative to finding a feasible solution.

Rubato-Opt has the fewest number of solutions. This is because at each iteration, much of the computation is driven by the objective function, which repeatedly produces candidates which violate the chance constraints. Rubato-Opt-FF, which tries to discover a first chance-constrained solution before further optimization, was able to solve more cases than Picard by itself. This demonstrates the importance of leveraging consistency checking to find negative cycles, which also help to guide optimization.

Recall that for each pSTN, the Rubato algorithms must allocate risk multiple times to find controllable STNUs. Risk allocation in these benchmarks is a convex nonlinear program (NLP), with the number of variables equal to twice the number of uncertain durations, since we are allocating risk to the tails of uncertain durations. The computation time spent in each round of risk allocation is thus a measure of the difficulty of risk allocation in each approach, and contribute to the total computation time.

Figure 16 shows, for each benchmark scenario, the average computation time for each round of risk allocation for the iterative methods. For the same number of uncertain durations, the optimizing approaches take significantly more time as expected. This is because during risk allocation, Rubato can stop as soon as a first feasible risk allocation is found, rather than finding the optimal risk allocation.

However, Rubato-Opt-FF, performs better slightly better than Rubato-Opt. This is because most of the nonlinear optimizations are essentially the same as that of Rubato, until the first feasible solution is found.

Each candidate risk allocation must be generated and the resulting STNU checked for controllability, with negative cycle conflicts extracted where the STNU is uncontrollable. The total computation time thus grow with the number of candidates generated.

Figure 17 compares the number of candidates generated by Rubato-Opt-FF and Rubato-Opt in scenarios solved by both algorithms. Similarly, we also compare the number of candidates generated by Rubato-Opt-FF against those generated by Rubato in Figure 18.

In comparing Rubato-Opt-FF against Rubato-Opt, we note that the number of candidates generated is similar, although Rubato-Opt-FF seems to require fewer risk allocations in general. This indicates that many of the constraints required to ensure an optimal solution can be collapsed by more concise constraints discovered during feasibility checking.

The number of candidate risk allocations generated for Rubato-Opt-FF is slightly higher than that of the Rubato. This is as expected since Rubato-Opt-FF essentially runs Rubato until a first feasible solution is found, after which it proceeds to optimize the risk allocation with respect to the objective function using Picard. Empirically the number of additional candidates generated to prove optimality is small. We thus only need to pay a small cost in the number of candidates generated to ensure optimality.

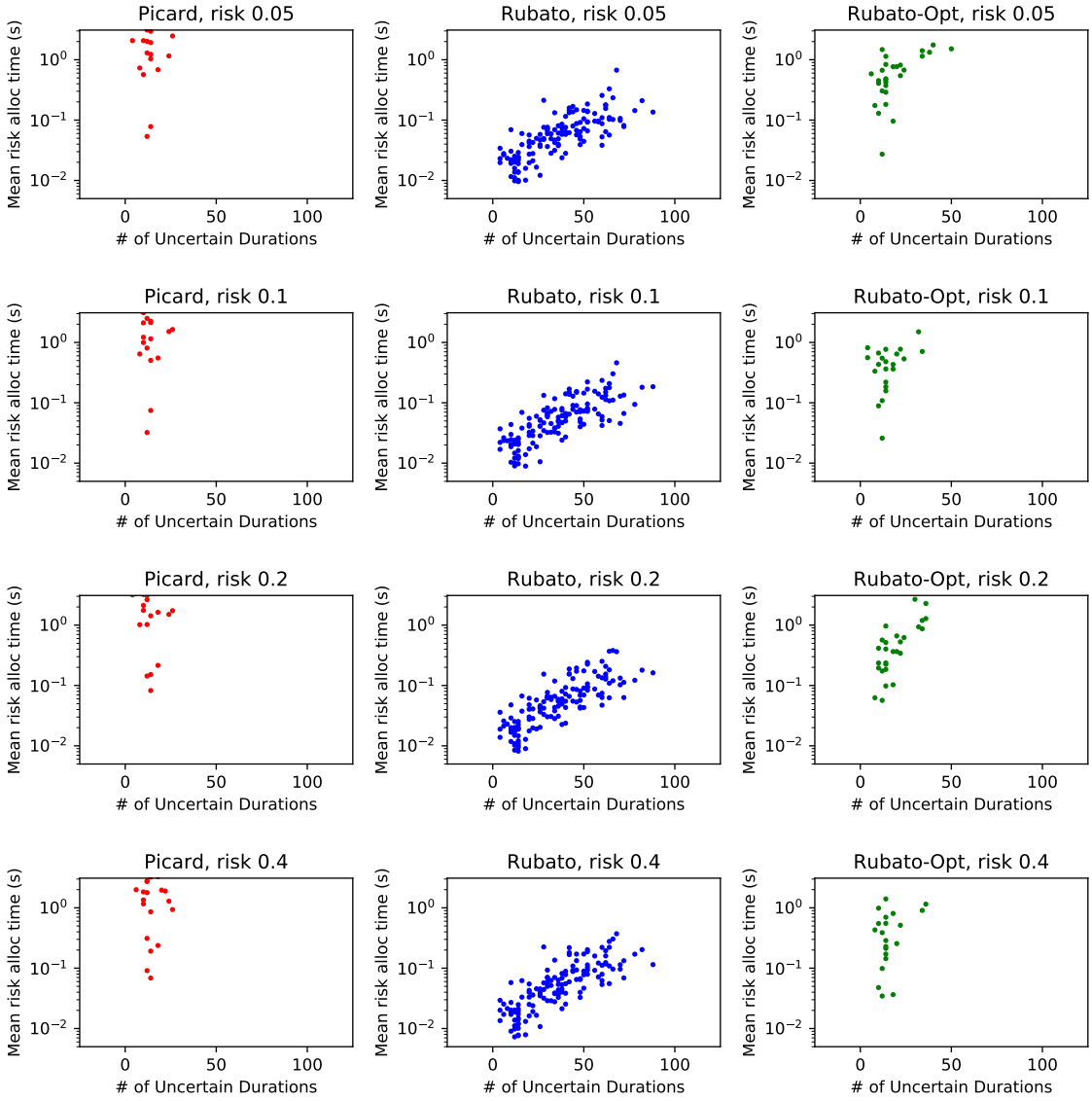


Figure 16: Average run time of risk allocation for benchmark scenarios.

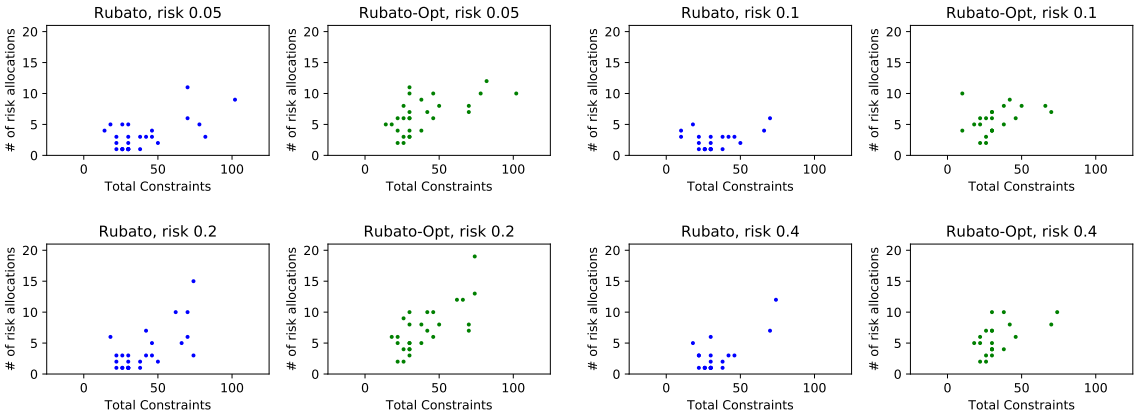


Figure 17: Number of candidate risk allocations where solutions were found for both Feasible-First and Objective-Throughout.

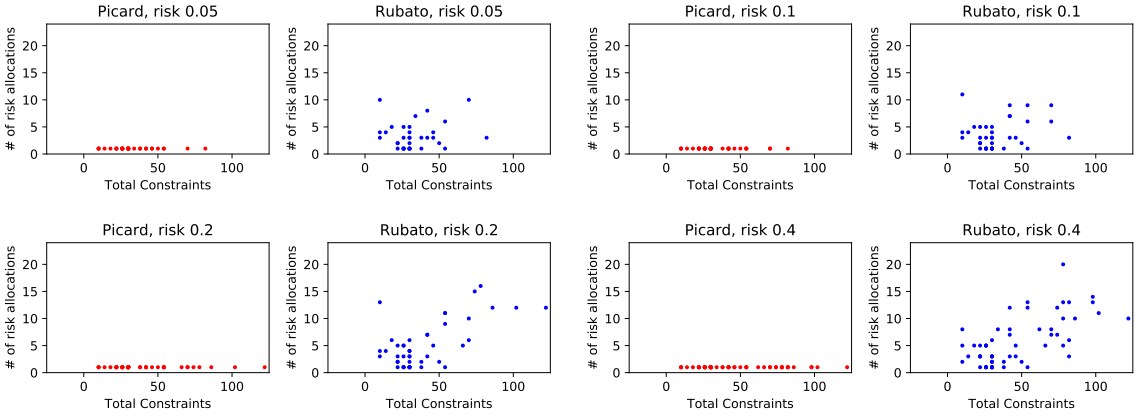


Figure 18: Number of candidate risk allocations where solutions were found for both Feasible-First and the consistency checking solution.

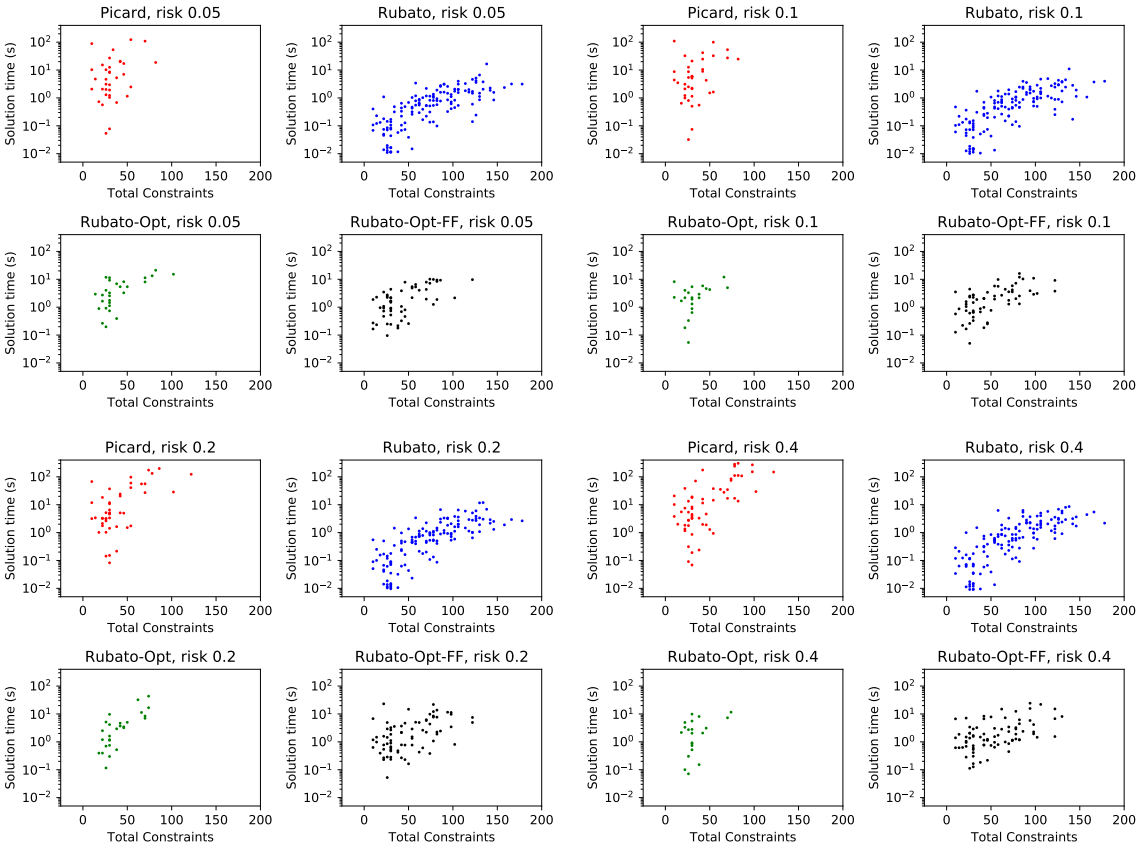


Figure 19: Total run times where solutions were found

Figure 19 shows the total runtimes required for solutions. As expected, Picard took far longer than the alternative algorithms. Of the iterative algorithms, the Rubato-Opt took the longest. This is expected, as we have seen above that the approach requires both a higher number of risk allocations, and has a higher average time spent in each risk allocation.

While Rubato-Opt-FF was slower than Rubato, the difference decreased as the risk bound loosened. This is because the number of candidates required to find the optimal risk allocation given a first feasible solution also decreased with a loosening risk bound, as in Figure 18. Intuitively, given the abundance of risk allowed with looser risk bounds, there is little to improve on a risk allocation derived from the first feasible solution. This is supported when considering the difference in quality of solutions with loosening risk bounds in Figure 15.

In summary, while Picard is able to provide optimal solutions to the approximate cc-pSTP problem, it is often slow in terms of runtime. On the other hand, Rubato is fast, but the solutions found are suboptimal. Rubato-Opt attempts to unite the two approaches by incorporating the objective function in the risk allocation steps of Rubato. However, this leads to a greater number of risk allocations, each of which is slower to complete. Rubato-Opt-FF, in which the objective function is only considered after a first feasible solution has been found, addresses the weaknesses of Rubato-Opt, allowing optimal solutions with computation speed comparable to Rubato.

## 7. Conclusion

This work has been motivated by the need to deal with uncertainty in scheduling. These are commonly encountered in applications from remote science exploration missions to transportation logistics. While prior representations concentrated on interval-bounded descriptions of uncertainty with the simple temporal problems with uncertainty (STNU) representation, we proposed the probabilistic simple temporal network (pSTN). The uncertain durations are described probabilistically and we gain flexibility by reasoning over the relative likelihood of different outcomes.

However, the use of the probabilistic representation of uncertainty allows for a potentially unbounded range of outcomes. Thus, instead of schedules guaranteed to be consistent with all temporal constraints for all outcomes of the uncertain durations, we must instead find schedules with guarantees over the probability of success. We thus proposed the chance-constrained probabilistic simple temporal problem (cc-pSTP). Instead of notions of controllability in prior work with interval-bounded uncertainty, we proposed chance-constrained controllability, such that the schedules are valid with a high probability. Intuitively, risk can be thought of as a resource in the cc-pSTP, and distributed for feasibility and higher utility solutions.

We noted first the difficulty of evaluating the probability of success given a schedule and a set of temporal constraints. However, we were able to propose an approximation of the cc-pSTP by reasoning over the marginals of each of the uncertain durations. The key insight was that, by allocating risk over each marginal, we were able to propose candidate interval-bounded uncertain durations. In effect, by good distribution of the allowed risk, we generate STNUs for which feasible schedules with high utility exist. This insight allows us to

encode the approximate cc-pSTP as a nonlinear program, with a risk allocation constraint in addition to standard constraints enforcing controllability from STNU literature.

We further noted that there is a natural decomposition of the nonlinear program. We may separate the problem into a nonlinear programming master, which enforces the chance-constraint and performs risk allocation over the uncertain durations, and a subsolver which checks the STNUs resulting from risk allocation. The subsolver can be implemented as negative cycle detection over a directed graph, meaning that checking can be performed efficiently. In addition, the negative cycles can be returned to guide subsequent risk allocations. This approach allows us to efficiently generate and test solutions, and perform risk allocation consistent with only the set of relevant constraints.

While the decomposition was able to find solutions which satisfied the constraints, the approach did not naturally consider the objective function. We have outlined two additional methods for incorporating the objective function: 1) we explicitly considering the objective function at every iteration of risk allocation; 2) we first find a feasible solution using the decomposition, and then use that as an initial point for the nonlinear optimization.

We have performed numerical experiments to compare the speed of the nonlinear optimization approach relative to the decomposition approach for finding as satisfying solution, as well as the two extensions.

The contributions of this work are summarized as follows:

1. The pSTN representation for temporal problems with probabilistic durations;
2. Formulation of chance-constrained controllability for pSTNs, and the cc-pSTP for scheduling with guarantees on consistency with temporal constraints;
3. An encoding for an approximate solution to the cc-pSTP via nonlinear programming;
4. A conflict-directed decomposition of the solution encoding leveraging efficient techniques for checking controllability;
5. Two extensions which combine the conflict directed and nonlinear programming approaches, allowing for faster computation of solution which also consider the objective function; and
6. A set of numerical experiments demonstrating how the extensions improve on the prior approaches, in terms of solution quality and solution speed.

### 7.1 Future work

There are three main avenues for future work: 1) extension beyond static schedules; 2) better cut generation; 3) multiple cuts for each risk allocation; and 4) a branch and bound approach.

In our current treatment of the chance-constrained simple temporal problem, we have focused on static scheduling of controllable events. However, the body of work investigating dynamic controllability provides insights into dynamic policies for execution in the presence of uncertain durations. One extension of the current work would consider how the decomposition approach could be applied to find chance-constrained policies, leveraging some of the insight in prior work. In particular, Cui et al. (2015) had insights into casting

restoring dynamic controllability as a mixed-integer linear program. One avenue of ongoing investigation is how we can leverage the insights in this paper to provide a customized branch-and-bound approach to solving the mixed problem.

We have also seen in the results that the risk allocation nonlinear optimization steps take up the majority of the runtime. If we consider each risk allocation as a chance to learn about the underlying structure of the simple temporal constraints, then we may be motivated to make the best use of each risk allocation.

The current iterative approach also does not discriminate between different negative cycles when generating linear constraints for risk allocation from the results of strong controllability checking. However, one negative cycles may result in a linear constraint that eliminates a larger portion of solution space than another, and may thus lead to faster convergence. One approach would be to appeal to the correspondence between negative cycles, and feasibility cuts in Bender's Decomposition. Future work may apply different selection criteria (Fischetti et al., 2010) to determine the best choice of negative cycles.

Alternatively, future work would concentrate on schemes for generating multiple negative cycles for each risk allocation. One such scheme is as follows: given an infeasible STN with controllability reductions, find a negative cycle, remove the edges comprising the negative cycle, and test the resulting STN for feasibility. By repeating this process, we are able to extract negative cycles which are independent of each other, and generate multiple linear constraints for each risk allocation.

Lastly, we can consider branch-and-bound techniques for the problem. Rather than only using negative cycles to generate linear cuts, and switching to optimization after the first feasible risk allocation is found, we can also consider ways of upper-bounding the cost of the feasible risk allocations. This may again be done by appealing to Bender's Decomposition. While STN checking can only give us feasibility cuts, we can consider the optimization problem with fixed bounds on contingent constraints, given a feasible risk allocation. For certain classes of objective functions, we may be able to find efficient subsolvers for the resulting numerical programs, and return optimality cuts as constraints to the risk allocation solver. These would allow us to summarize how future risk allocations may be performed to give better candidates.

## Appendix A. STNU Strong Controllability Reductions

The check for strong controllability outlined in Vidal and Fargier (1999) is a proof by construction: the algorithm attempts to construct a schedule which will work for all possible values for the uncertain durations. This is especially powerful, as we will not only be able to determine whether the problem is strongly controllable, but also obtain a schedule which will work under all circumstances if the problem is strongly controllable. The key insight which allowed this is the fact that we can override requirement constraints involving uncontrollable events with new requirement constraints over controllable events. This turns the STNU into a STN, for which polynomial solution methods are known. Before outlining the rationale, we walk through an example.

**Example 11.** *Consider the STNU as seen in Figure 20. We have two controllable events, and one uncontrollable event. The uncontrollable event $t_\omega$ occurs in the interval $[c, d]$ after*

*the execution of $t_0$, and we must schedule $t_1$ and $t_0$ such that $t_1$ occurs between $[a, b]$ before $t_\omega$.*

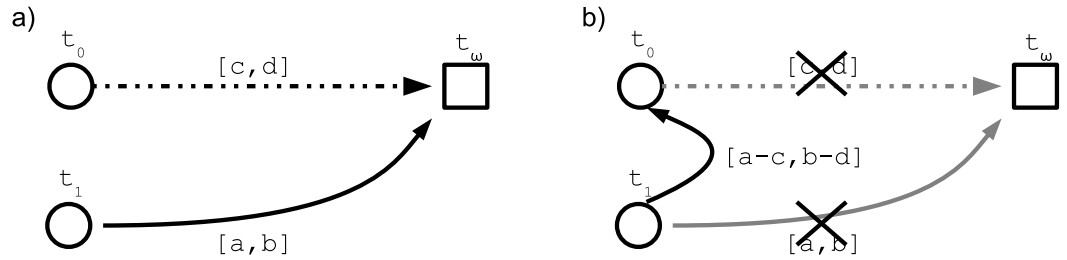


Figure 20: Example of how to override constraints involving uncontrollable event.

*Let us consider the constraints between $t_0$ and $t_1$, and see if we can obtain some constraint $t_0 - t_1 \in [a', b']$. As a first step, observe that $t_\omega$ may be written $t_0 + \omega$, where $\omega$ is some value between $[c, d]$ assigned by the environment. Then, we can write the constraint between $t_\omega$ as*

$$t_0 + \omega - t_1 \in [a, b]$$

*Consider first the lower bound, that is, $t_0 + \omega - t_1 \geq a$. Note that, this is guaranteed for all values of $\omega \in [c, d]$ iff $t_0 - t_1 \geq a - c$. Consider now the upper bound, $t_0 + \omega - t_1 \leq b$, or equivalently $t_1 - t_0 - \omega \geq -b$. Note that this is guaranteed if and only if $t_1 - t_0 \geq -b - d$. We have thus found a set of constraints equivalent to those in the original problem, and we may replace the original constraints with the new constraint, removing the uncontrollable event and the associated contingent constraint. This conveniently leaves us with a STN, shown on the right in Figure 20.*

The above demonstrates how we may reduce away the contingent constraints in an STNU for strong controllability. The process can be generalized, such that any requirement constraint to an uncontrollable event may be replaced by requirement constraints between controllable events. The new constraints allow us to reason without dealing directly with the uncontrollable constraints. The problem is reduced from a game against an uncooperative environment to a standard constrained optimization problem solvable with readily available packages.

We repeat the general reductions from Vidal and Fargier (1999) to develop our algorithm. Consider STNU $\mathcal{N}^u = \langle E^c, E^u, C^r, C^c \rangle$. Let $C^{r-} \subseteq C^r$ be the set of requirement constraints involving uncontrollable events. These are the only constraints which depend on variables not controlled by the agent, and thus need to be reframed.

We consider the lower and upper bounds separately and first perform our analysis for lower bounds. For each lower bound $c \in C^{r-}$, we obtain one of three cases in Figure 21:

**Case 1** (only the end event is uncontrollable) $t_i + \omega_i - t_j \geq a$

**Case 2** (only the start event is uncontrollable) $t_i - t_j - \omega_j \geq a$

**Case 3** (both start and end are uncontrollable) $t_i + \omega_i - t_j - \omega_j \geq a$

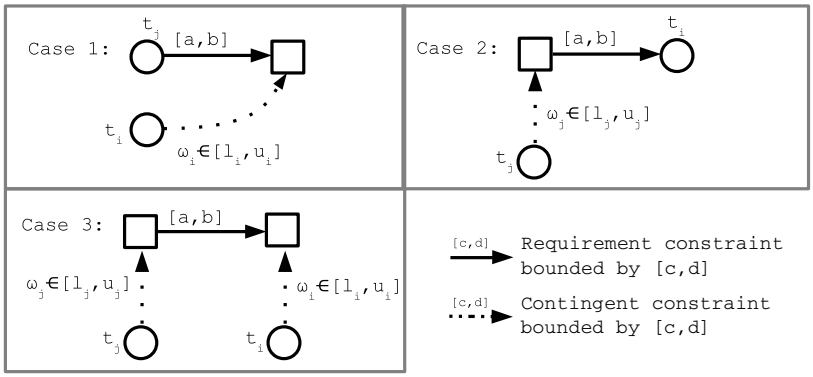


Figure 21: Three cases for reductions.

where $\omega_i$ is the duration of the contingent constraint starting at controllable event $t_i$ and bounded by $[l_i, u_i]$, with $\omega_j$ similarly defined.

In each case, we replace the original requirement constraint with a new requirement constraint:

**Case 1** $t_i - t_j \geq a - l_i$

**Case 2** $t_i - t_j \geq a + u_j$

**Case 3** $t_i - t_j \geq a - l_i + u_j$

The corresponding reductions for the upper bounds can be written similarly by multiplying both sides by $-1$.

In this way, an STNU may be rewritten with only requirement constraint between controllable events. We may discard the contingent constraints because the constraints are already encoded by the reformulation outlined above. We may thus obtain an STN with only the requirement constraints and the controllable events. A consistent solution to the STN *will be a schedule for the STNU valid under any combination of outcomes for the contingent constraints*, because the constraints derived from the contingent constraints require consistency for all possible outcomes. More formally, let $\widehat{C}^{r-}$ be the collection of requirement constraints obtained from the reduction.

**Theorem 3.** *If there exists a schedule $S$ satisfying all constraints in $(C^r \setminus C^{r-}) \cup \widehat{C}^{r-}$, then the STNU is* strongly controllable*. Further, given any combination of outcomes for $C^c$ the set of contingent constraints, $S$ is consistent with respect to all elements in $C^r$ the set of requirement constraints.*

The interested reader may be referred to the more detailed exploration in Vidal and Fargier (1999). However, the above allows robust scheduling. If the uncertain durations are interval-bounded such that the corresponding STNU is strongly controllable, then we can schedule the activated time points to be temporally consistent for every possible outcome of the uncertain durations.

## Appendix B. Negative Cycles as Bender's Feasibility Cuts

In this section, we show that negative cycles have a natural interpretation as feasibility cuts in a Generalized Bender's Decomposition (Benders, 1962; Geoffrion, 1972) framework.

In a Bender's Decomposition framework, a larger problem can be decomposed into two steps. The candidate generation step suggests potential assignments to a subset of the decision variables, the choice of which allows the remaining constraints and decision variables to be considered as a subproblem for which we can apply efficient solvers. This is applicable to the cc-pSTP framework among others such as those in Yu et al. (2015); Cui et al. (2015), as these have subproblems which can be formulated as simple temporal problems. Key to the efficiency of such approaches is the derivation of Bender's feasibility cuts given infeasible candidates, which help the generator to avoid generating further infeasible candidates. These cuts are generated using unbounded rays for the dual of the subproblem.

We will show that negative cycles correspond to a special case of unbounded rays, which arise when the subproblem can be formulated as a system of difference constraints, as in the case of simple temporal problems. We will do so by first noting that the simple temporal problem may be solved via negative cycle detection on a distance graph. We then present the dual problem of the numerical program encoded by the distance. Lastly we show that the unbounded rays can be constructed using the edges involved in a negative cycle.

A set of simple temporal constraints may be mapped to a distance graph representing a set of difference constraints, with decision variables corresponding to assignments to controllable events (Dechter et al., 1991). The negative cycle detection algorithm thus solves the following numerical program:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}} \quad & 0 \\ st. \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned} \tag{B.1}$$

where:

- $\mathbf{x}$ is a set of real decision variables. These represent the assignments to the controllable events, which are the nodes in the distance graph;
- $\mathbf{A}$ is a matrix such that every row has two elements, one of which has value -1, and the other has value 1. These correspond to the two controllable events involved in each simple temporal constraint, which are respectively the source and sink in the distance graph; and
- $\mathbf{b}$ is a vector of real numbers. These represent the upper-bound of each temporal constraint, which are the edge distances for each edge of the distance graph.

Taking the standard linear dual, we have the dual problem:

$$\begin{aligned} \max_{\mathbf{u}\geq\mathbf{0}} \quad & -\mathbf{b}^T\mathbf{u} \\ st. \quad & \mathbf{A}^T\mathbf{u} = \mathbf{0} \end{aligned}$$

where $\mathbf{u}$ is the vector of Lagrangian multipliers. Each element of $\mathbf{u}$ corresponds to a constraint in the primal problem, and thus an edge in the distance graph. Note that this dual

problem is equivalent to

$$\begin{aligned} \min_{\mathbf{u} \geq \mathbf{0}} \quad & \mathbf{b}^T \mathbf{u} \\ st. \quad & \mathbf{A}^T \mathbf{u} = \mathbf{0} \end{aligned} \tag{B.2}$$

Suppose that the primal for the subproblem given in B.1 is infeasible. In a Bender's decomposition framework, the dual given in B.2 would be unbounded, and we must construct a feasibility cut using the unbounded ray $\mathbf{u}^*$, a vector along which the dual is unbounded.

However, if the primal problem is infeasible, we can construct a negative cycle on the distance graph. For a negative cycle, let us construct a ray $\mathbf{u}_{neg}$, such that the $i$th element of $\mathbf{u}_{neg}$ is 1 if the corresponding edge features in the negative cycle, and 0 otherwise. We can verify that $\mathbf{u}_{neg}$ is an unbounded ray for the dual given in B.2.

Note first that $\mathbf{A}^T \mathbf{u}_{neg} = \mathbf{0}$, because $\mathbf{u}_{neg}$ correspond to edges in a cycle. Consider the indices of the non-zero elements of $\mathbf{u}_{neg}$ - these correspond to the edges in the negative cycle. Each row in $\mathbf{A}^T$ correspond to edges associated with a node in the distance graph, such that for the element $\mathbf{A}^T_{ij}$ at row $i$ and column $j$, $\mathbf{A}^T_{ij}$ is:

- 1 if edge $j$ is incoming to node $i$;
- -1 if edge $j$ is outgoing from node $i$; and
- 0 otherwise.

By definition, for every node in a cycle, the number of incoming edges is equal to the number of outgoing edges. Thus, for every row $\mathbf{A}^T_i$ of $\mathbf{A}^T$, we have $\mathbf{A}^T \mathbf{u}_{neg} = 0$.

Further, note that $\mathbf{b}$ is the list of edge weights for the distance graph. By construction, $\mathbf{u}_{neg} \geq \mathbf{0}$, and $\mathbf{b}^T \mathbf{u}_{neg} = \psi < 0$. Thus, for $\lambda > 1$, $\mathbf{b}^T[\lambda \mathbf{u}_{neg}] = \lambda \psi < 0$.

Thus, B.2 is unbounded and $\mathbf{u}_{neg}$ is an unbounded ray, as $\lambda \mathbf{u}_{neg}$ meets the constraints, for arbitrarily large values of $\lambda$, and results in an objective value that is arbitrarily negative.

This shows that negative cycles correspond to feasibility cuts in a Bender's Decomposition framework, in the special case when the subproblem is one of satisfying a set of difference constraints. This motivates the use of negative cycles to summarize infeasibility in our work and related work where the subproblem can be expressed as checking feasibility for a system of simple temporal constraints.

## References

Beck, J. C. and Wilson, N. (2007). Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research*, 28:183–232.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4(1):238–252.

Blackmore, L., Ono, M., Bektassov, A., and Williams, B. C. (2010). A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *Robotics, IEEE Transactions on*, 26(3):502–517.

Blackmore, L. and Williams, B. C. (2007). Optimal, robust predictive control of nonlinear systems under probabilistic uncertainty using particles. In *American Control Conference, 2007. ACC'07*, pages 1759–1761. IEEE.

Charnes, A. and Cooper, W. W. (1959). Chance-constrained programming. *Management science*, 6(1):73–79.

Conrad, P. R. and Williams, B. C. (2011). Drake: An efficient executive for temporal plans with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659.

Cui, J., Yu, P., Fang, C., Haslum, P., and Williams, B. C. (2015). Optimising bounds in simple temporal networks with uncertainty under dynamic controllability constraints. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*.

De Kleer, J. and Williams, B. C. (1989). Diagnosis with behavioral modes. In *IJCAI*, volume 89, pages 1324–1330.

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85.

Dechter, R., Meiri, I., and Pearl, J. (1991). Temporal constraint networks. *Artificial intelligence*, 49(1):61–95.

Fang, C., Yu, P., and Williams, B. (2014). Chance-constrained probabilistic simple temporal problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Fischetti, M., Salvagnin, D., and Zanette, A. (2010). A note on the selection of benders' cuts. *Mathematical Programming*, 124(1):175–182.

Fu, N., Lau, H. C., Varakantham, P., and Xiao, F. (2012). Robust local search for solving rcpsp/max with durational uncertainty. *Journal of Artificial Intelligence Research*, 43:43–86.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260.

Gill, P. E., Murray, W., and Saunders, M. A. (2005). Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131.

Hnich, B., Rossi, R., Tarim, S. A., and Prestwich, S. (2012). Filtering algorithms for global chance constraints. *Artificial Intelligence*, 189:69–94.

Hunsberger, L., Posenato, R., and Combi, C. (2012). The dynamic controllability of conditional stns with uncertainty. *PlanEx 2012*.

Khatib, L., Morris, P., Morris, R., Rossi, F., et al. (2001). Temporal constraint reasoning with preferences. In *IJCAI*, pages 322–327.

Kim, P., Williams, B. C., and Abramson, M. (2001). Executing reactive, model-based programs through graph-based temporal planning. In *IJCAI*, pages 487–493.

Lau, H. C. and Hoang, T. A. (2014). Risk minimization of disjunctive temporal problem with uncertainty. *Knowledge and Systems Engineering*, pages 223–236.

Lau, H. C., Li, J., and Yap, R. H. (2006). Robust controllability of temporal constraint networks under uncertainty. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 288–296. IEEE.

Levine, S. J. and Williams, B. C. (2018). Watching and acting together: Concurrent plan recognition and adaptation for human-robot teams. *Journal of Artificial Intelligence Research*, 63:281–359.

Lund, K., Dietrich, S., Chow, S., and Boerkoel, J. (2017). Robust execution of probabilistic temporal plans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Mendes, S., Duncombe, R., Scarlett, R., Shaffer, J., Lensch, S., and Valentine, D. (2013). Microbial oxidation of ethane within seep sediment at coal oil point, santa barbara, ca. In *AGU Fall Meeting Abstracts*.

Miller, B. L. and Wagner, H. M. (1965). Chance constrained programming with joint constraints. *Operations Research*, 13(6):930–945.

Morris, P. and Muscettola, N. (2005). Temporal dynamic controllability revisited. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005*, pages 1193–1198. AAAI Press / The MIT Press.

Nemirovski, A. and Shapiro, A. (2006). Scenario approximations of chance constraints. *Probabilistic and randomized methods for design under uncertainty*, pages 3–47.

Nemirovski, A. and Shapiro, A. (2007). Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996.

Ono, M. and Williams, B. (2008). Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3427–3432. IEEE.

Ono, M., Williams, B. C., and Blackmore, L. (2013). Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577.

Saint-Guillain, M., Stegun Vaquero, T., Agrawal, J., and Chien, S. (2020). Robustness computation of dynamic controllability in probabilistic temporal networks with ordinary distributions. In *Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*.

Saint-Guillain, M., Vaquero, T., Chien, S., Agrawal, J., and Abrahams, J. (2021). Probabilistic temporal networks with ordinary distributions: Theory, robustness and expected utility. *Journal of Artificial Intelligence Research*, 71:1091–1136.

Stallman, R. M. and Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial intelligence*, 9(2):135–196.

Stergiou, K. and Koubarakis, M. (2000). Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117.

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

Tsamardinos, I., Pollack, M. E., and Ramakrishnan, S. (2003). Assessing the probability of legal execution of plans with temporal uncertainty. In *Proc. of ICAPS'03 Workshop on Planning Under Uncertainty and Incomplete Information*.

Van Hessem, D. and Bosgra, O. (2006). Stochastic closed-loop model predictive control of continuous nonlinear chemical processes. *Journal of Process Control*, 16(3):225–241.

Venable, K. B. and Yorke-Smith, N. (2005). Disjunctive temporal planning with uncertainty. In *IJCAI*, pages 1721–1722.

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

Walsh, T. (2002). Stochastic constraint programming. In *ECAI*, volume 2, pages 111–115.

Wang, A. J. and Williams, B. C. (2015). Chance-constrained scheduling via conflict-directed risk allocation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Yorke-Smith, N., Venable, K., and Rossi, F. (2003). Temporal reasoning with preferences and uncertainty. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1385–1386.

Yu, P., Fang, C., and Williams, B. (2015). Resolving over-constrained probabilistic temporal problems through chance constraint relaxation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

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