Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

PARIS: A Polynomial-Time, Risk-Sensitive Scheduling Algorithm for Probabilistic Simple Temporal Networks with Uncertainty

Pedro Santana,[*] **Tiago Vaquero,**[*†] **Cláudio Toledo,**[+]
Andrew Wang,[*] **Cheng Fang,**[*] **and Brian Williams**[*]

[*]MIT, CSAIL
Cambridge, MA 02139
{psantana,tvaquero,wangaj,cfang,williams}@mit.edu

[+]USP, ICMC
São Carlos, SP, Brazil
claudio@icmc.usp.br

† Caltech
Pasadena, CA 91125
tstegunv@caltech.edu

Abstract

Inspired by risk-sensitive, robust scheduling for planetary rovers under temporal uncertainty, this work introduces the Probabilistic Simple Temporal Network with Uncertainty (PSTNU), a temporal planning formalism that unifies the set-bounded and probabilistic temporal uncertainty models from the STNU and PSTN literature. By allowing any combination of these two types of uncertainty models, PSTNU's can more appropriately reflect the varying levels of knowledge that a mission operator might have regarding the stochastic duration models of different activities. We also introduce PARIS, a novel sound and provably polynomial-time algorithm for risk-sensitive strong scheduling of PSTNU's. Due to its fully linear problem encoding for typical temporal uncertainty models, PARIS is shown to outperform the current fastest algorithm for risk-sensitive strong PSTN scheduling by nearly four orders of magnitude in some instances of a popular probabilistic scheduling dataset, while results on a new PSTNU scheduling dataset indicate that PARIS is, indeed, amenable for deployment on resource-constrained hardware.

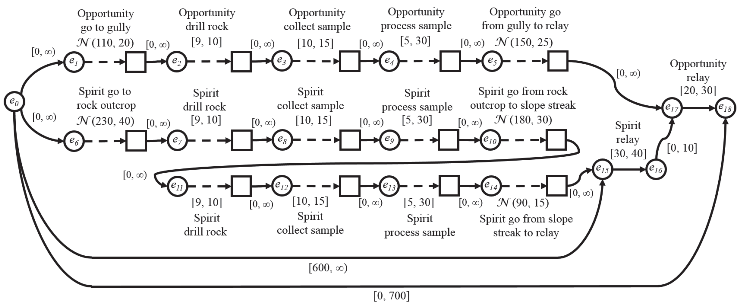
1 Introduction

Endowing autonomous agents with a keen sensitivity to temporal uncertainty is key in enabling them to reliably complete time-critical missions in the real-world. Such requirement has been the subject of recent research initiatives to incorporate duration uncertainty into temporal planning frameworks (Beaudry, Kabanza, and Michaud 2010; Cimatti, Micheli, and Roveri 2015; Micheli, Do, and Smith 2015). A dominant effort has been to extend the set-bounded uncertainty model from Simple Temporal Networks with Uncertainty (STNU's) (Vidal 1999) to a probabilistic setting, in which the risk of violating deadlines and other temporal requirements can be quantified (Tsamardinos 2002).

Three essential questions about Probabilistic Simple Temporal Networks (PSTN's) have been previously addressed. Tsamardinos (2002) showed that a risk-minimizing schedule of a PSTN could be derived analytically. In his derivation, he reduced the PSTN to an STNU with variable bounds, and effectively solved for the bounds that would guarantee strong controllability. This reduction was further developed in the context of chance constraints (Birge and Louveaux 1997), where one seeks a schedule whose risk of violating temporal requirements is guaranteed to be bounded by a user-defined threshold. Operating in a general nonlinear optimization setting, Fang, Yu, and Williams (2014) applied the STNU strong controllability reductions from (Vidal 1999) to propose *Picard*, a chance-constrained strong scheduler for PSTN's capable of optimizing any schedule-related objective function. Wang and Williams (2015), on the other hand, present *Rubato*, an efficient algorithm for verifying the existence of strong chance-constrained schedules that leverages an SMT architecture to propose candidate STNU's that meet the chance constraints and discover temporal conflicts.

Although previous work introduces effective methods for dealing with temporal uncertainty and scheduling risk, two main challenges remain. First, neither STNU's, nor PSTN's, accurately capture the varying levels of knowledge that a mission operator might have about the sources of temporal uncertainty that can have an impact on a mission. While set-bounded uncertainty in STNU's fails to represent frequency models for random quantities captured by probability distributions, the fully probabilistic models in a PSTN assumes a potentially unreasonable level of understanding of the natural phenomena that give rise to uncertain temporal behavior. Therefore, a first contribution of this work is to unify STNU and PSTN models under Probabilistic Simple Temporal Networks with Uncertainty (PSTNU's).

Our second and most important contribution is related to the complexity of scaling current probabilistic scheduling methods, especially if one has in mind the goal of running such algorithms on embedded robotic hardware, where on-board computation and energy is scarce. While the state of the art invariably resorted to general-purpose nonlinear solvers to implement probabilistic scheduling methods, this work introduces the Polynomial-time Algorithm for RIsk-aware Scheduling (PARIS), a new and provably polynomial-time algorithm for strong scheduling of PSTNU's. Inspired by linear approximation techniques used to reduce the complexity of AC power flow analysis (Coffrin and Van Hentenryck 2014), PARIS leverages a *fully linear* encoding of the risk-aware probabilistic scheduling problem that allows it to drastically reduce runtime and memory requirements. Empirical evaluation on a probabilistic scheduling dataset



(a) Probabilistic Simple Temporal Network with Uncertainty (PSTNU).

e_0=0	e_1=0	e_2=132.35
e_3=142.35	e_4=157.35	e_5=187.35
e_6=0	e_7=246.64	e_8=256.64
e_9=271.64	e_{10}=301.64	e_{11}=495.75
e_{12}=505.75	e_{13}=520.75	e_{14}=550.75
e_{15}=650	e_{16}=680	e_{17}=680
e_{18}=700		

(b) Activity schedule computed by the Polynomial-time Algorithm for RIsk-aware Scheduling (PARIS).

Figure 1: Rover coordination under temporal uncertainty. (a) Scenario representation as PSTNU. (b) Strong activity schedule for the PSTNU in (a) with scheduling risk bound of 6.7% (i.e., all temporal requirements met with probability of at least 93.3%).

first introduced in (Fang, Yu, and Williams 2014) shows a several-order-of-magnitude improvement over the current fastest algorithm for PSTN scheduling, while our new planetary rover-inspired PSTNU dataset indicates that PARIS is, indeed, amenable for deployment on embedded hardware.

1.1 Motivation: planetary rover coordination

Future planetary exploration missions will require an increasing level of coordination between multiple spacecrafts under temporal uncertainty. Figure 1 depicts a planetary exploration scenario illustrating important requirements for such missions: two autonomous rovers, *Spirit* and *Opportunity*, should explore three sites on a region of Mars (gully, slope streak, and rock outcrop). After exploring all sites, both rovers must travel to a relay site and transmit their findings to an orbiting satellite within a limited time window.

There are several complicating temporal factors. First, traversal times between locations are uncertain. However, rover mobility has been extensively studied, both from experience on Mars and in simulated environments. Therefore, these times can be modeled probabilistically based on the distance between sites and terrain features. Second, the science-gathering activities at each site have uncertain durations. Due to lack of real and simulated data, there are no uncertainty models to aid in the prediction of how long these activities will last, but lower and upper bounds are built into the rover firmware to prevent deadlocks. Finally, at the relay site, there is an absolute time window in which the satellite is in field-of-view. Seeking to maximize throughput, both rovers should transmit at approximately the same time, but their transmissions should not overlap.

Due to power and communication limitations, the rovers cannot coordinate during plan execution. Therefore, one needs to precompute a schedule that satisfies all temporal requirements, while being robust to the uncertainty in activity durations. In other words, this is a strong temporal planning problem, which Micheli, Do, and Smith (2015) note is applicable to many safety-critical autonomous missions. The work of Cimatti, Micheli, and Roveri (2015) approaches strong temporal planning by replacing the STN scheduler in COLIN (Coles et al. 2012) with an STNU strong controllability solver. In our scenario, we would like a risk-aware strong controllability solver that handles both probabilistic and set-bounded uncertain durations, which could be incorporated into a planner in a similar manner.

2 Background & PSTNU's

Motivated by the aforementioned rover coordination scenario, our risk-aware scheduling methods operate on Probabilistic Simple Temporal Networks with Uncertainty (PSTNU's), a novel temporal modeling formalism unifying features from Simple Temporal Networks (STN's) (Dechter, Meiri, and Pearl 1991), Simple Temporal Networks with Uncertainty (STNU's) (Vidal 1999), and Probabilistic Simple Temporal Networks (PSTN's) (Tsamardinos 2002). For the sake of completeness, we briefly review these concepts. Figure 2 shows the different elements in a PSTNU.

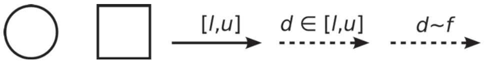


Figure 2: Elements of a PSTNU, where $[l, u]$ is a given interval and f is a known probability density function (pdf). From left to right: controllable event; contingent (uncontrollable) event; STC; STCU; PSTC.

An STN (Definition 1) can model scheduling problems where the agent has control over the temporal assignments (a.k.a. the *schedule*) to all events. Possible assignments are restricted by simple temporal constraints (STC's), which limit the distance between the timing of two events. If there exists at least one schedule fulfilling all STC's, we say that the STN is *consistent*. Otherwise, it is *inconsistent*.

Definition 1 (STN). *An STN is a tuple $\langle \mathcal{E}_c, \mathcal{C}_r \rangle$, where*

- $\mathcal{E}_c$ *: set of **controllable** temporal events, all of which must be assigned by the scheduler;*
- $\mathcal{C}_r$ *: set of **requirement** STC's of the form $l \le e_2 - e_1 \le u$, $l, u \in \mathbb{R} \cup \{-\infty, \infty\}$, where $e_1, e_2 \in \mathcal{E}_c$.*

An STN is capable of modeling externally-imposed temporal requirements (e.g., "return to base in less than 30 minutes") and durations of agent-controlled activities (e.g., "hibernate for 1 hour"), but is unable to model activities with uncertain durations (e.g., "drill a 2 cm hole in a rock" or "travel between two sites"). An STNU (Definition 2) addresses this limitation by extending STN's with *contingent* (also called *uncontrollable*) constraints and events. In an STNU, a contingent constraint is represented as an STC with uncertainty (STCU), which allows the difference between two temporal events to be non-deterministic, but bounded by a known interval $[l, u]$. Depending on how much information about contingent durations is made available to the scheduler during execution, Vidal (1999) defines different levels of *controllability* for STNU's: *weak controllability* assumes all values of contingent durations to be known to the scheduler before it has to make any decisions; *strong controllability* assumes that no such information is ever available to the scheduler; and *dynamic controllability* assumes that the scheduler can only use information about past contingent durations when making future scheduling decisions.

Definition 2 (STNU)**.** *An STNU is a tuple $\langle \mathcal{E}_c, \mathcal{E}_u, \mathcal{C}_r, \mathcal{C}_u \rangle$ that extends STN's by adding:*

- $\mathcal{E}_u$ *: set of* ***contingent*** *temporal events, which are assigned by an uncontrollable external agent ("Nature");*
- $\mathcal{C}_u$ *: set of* ***contingent*** *simple temporal constraints with uncertainty (STCU's) of the form*

$$e_2 = e_1 + d, \;\; l \leq d \leq u, l, u \in \mathbb{R}_{>0}, \qquad (1)$$

where d is a set-bounded, non-deterministic duration; $e_1 \in \mathcal{E}_c \cup \mathcal{E}_u$; and $e_2 \in \mathcal{E}_u$.

- $\mathcal{C}_r$ *: same as in Definition 1, but we allow $e_1, e_2 \in \mathcal{E}_c \cup \mathcal{E}_u$.*

Modeling contingent durations using STCU's is rather restrictive, for they do not incorporate information about the relative frequency of the different values for d in (1). For instance, they cannot model the statement "the rover requires 20 minutes to complete the traversal on average, with a standard deviation of 5 minutes". To address this need, a PSTN (Definition 3) allows contingent durations to be represented as random variables with known probability distributions. The notions of controllability for STNU's can be readily transferred to PSTN's. Moreover, controllability for PSTN's can be further extended with a notion of *scheduling risk* (Tsamardinos 2002; Fang, Yu, and Williams 2014; Wang and Williams 2015).

Definition 3. *(PSTN) Similar to STNU's, a PSTN is a tuple $\langle \mathcal{E}_c, \mathcal{E}_u, \mathcal{C}_r, \mathcal{C}_u \rangle$, where $\mathcal{C}_u$ contains probabilistic simple temporal constraints (PSTC's). A PSTC is also of the form $e_2 = e_1 + d$, with the additional assumption that d is a continuous random variable following a known probability distribution with positive support.*

While STNU's ignore extra knowledge that one might have about temporal uncertainty, PSTN's can err on the side of requiring *too much knowledge* to be available. As exemplified in Section 1.1, probabilistic models may not be known for every source of uncertainty affecting a mission. In most practical applications, these models are obtained through statistical analysis of experimental data, and "guessing" a model in the absence of data can lead to unquantifiable levels of mission risk. A PSTNU (Definition 4) addresses these issues by simply allowing the uncertainty models from STNU's and PSTN's to co-exist, so that mission operators can leverage probabilistic models for temporal uncertainty if, and only if, there is evidence to support them.

Definition 4. *(PSTNU) Same as a PSTN, but $\mathcal{C}_u$ may contain any combination of STCU's and PSTC's.*

3 Problem formulation

Following (Vidal 1999), given $A \subseteq \mathcal{E}_c$ and $B \subseteq \mathcal{E}_u$, denote a *control sequence* $\delta : A \rightarrow \mathbb{R}$ and a *situation* $\omega : B \rightarrow \mathbb{R}$. Intuitively, δ represents a partial assignment to controllable events in a PSTNU, while ω is a partial assignment to contingent events made by Nature. If δ and ω assign values to every controllable and contingent event in a PSTNU, we call them *complete* control sequence and situation. Consider now the set of control sequences $\mathcal{S} = \{\delta \,|\, \delta : A \subseteq \mathcal{E}_c \rightarrow \mathbb{R}\}$ and the set of situations $\mathcal{O} = \{\omega \,|\, \omega : B \subseteq \mathcal{E}_u \rightarrow \mathbb{R}\}$. A *scheduling policy* $\pi : \mathcal{S} \times \mathcal{O} \times \mathcal{E}_c \rightarrow \mathbb{R}$ defines a strategy to schedule controllable events based on an adopted control sequence and the resulting situation. Due to uncertain durations, there might be a non-zero probability of a scheduling policy π violating requirement constraints in a PSTNU, giving rise to the notion of *scheduling risk* (Definition 5).

Definition 5. *(Scheduling risk) Let π be a scheduling policy for a PSTNU N, and $C \subseteq \mathcal{C}_r$. The scheduling risk for the pair $\langle \pi, C \rangle$ corresponds to the probability of complete control sequences generated by π violating one or more constraints in C.*

One should notice that Definition 5 considers violations of subsets of $\mathcal{C}_r$, the *requirement* constraints, but not $\mathcal{C}_u$. This is because contingent durations violating their corresponding STCU or PSTC are instances of modeling errors, a source of uncertainty outside the scope of PSTNU's.[1] For that reason, we henceforth assume that all contingent durations take place according to their modeled behavior in a PSTNU.

3.1 Computing strong schedules

Following (Tsamardinos 2002; Fang, Yu, and Williams 2014; Wang and Williams 2015; Cimatti, Micheli, and Roveri 2015; Micheli, Do, and Smith 2015), we focus our attention on *strong controllability* in Definition 6.

Definition 6. *(Strongly controllable PSTNU) A PSTNU N is strongly controllable (SC) if, and only if, there exists a complete control sequence δ such that, for all complete situations ω, all requirement constraints $\mathcal{C}_r$ in N are satisfied.*

A PSTNU N is strongly controllable if one can compute a complete assignment to controllable events regardless of the contingent duration values during execution, and still be guaranteed to fulfill all requirement constraints in N. We shall call such a schedule a *strong policy* π_s. Vidal (1999) introduces rules for checking strong controllability of STNU's for the particular case where $e_1 \in \mathcal{E}_c$ in (1), which were later applied to PSTN's by Fang, Yu, and Williams (2014). In support of the scheduling risk discussion in Section 3.2 and our polynomial-time scheduling algorithm in Section 4,

[1] Considering modeling errors can be useful, should one want to model uncertainty about the uncertainty models themselves.

we now derive linear programming-based necessary and sufficient conditions for PSTNU strong controllability for the general case where e_1 in (1) can be controllable or contingent. For that, let $e_2-e_1 \leq u$, $e_2-e_1 \geq l$, represent a generic requirement constraint c_r in a PSTNU N, where e_1 and e_2 can be either controllable or contingent events. A strong policy π_s assigns values to elements of $\mathcal{E}_c$, therefore rendering the satisfaction of c_r a function only of the contingent durations. Let Ω be the set of all possible complete situations, and Ω_i be the subset of a complete situation affecting event e_i (directly or indirectly). From Definition 6, a PSTNU N is strongly controllable if, and only if,

$$\forall c_r \in \mathcal{C}_r, \max_{\Omega}(e_2 - e_1) \leq u, \;\; \min_{\Omega}(e_2 - e_1) \geq l, \qquad (2)$$

where e_1 and e_2 are taken from c_r. To compute (2), use (1) to write

$$e_1 = e_1^c + \sum_{\Omega_1} d_i, \;\; e_2 = e_2^c + \sum_{\Omega_2} d_i, \qquad (3)$$

where $e_i^c \in \mathcal{E}_c$ is a controllable event fixed by the strong schedule, and Ω_i is the "contingent path" of e_i (for controllable e_i, $\Omega_i = \varnothing$ and $e_i = e_i^c$). In the difference $e_2 - e_1$, common contingent durations d_i cancel out, leaving

$$\max_{\Omega}(e_2-e_1) = e_2^c-e_1^c+ \sum_{\Omega_2\backslash\Omega_1} \max(d_i) - \sum_{\Omega_1\backslash\Omega_2} \min(d_i),$$

$$\min_{\Omega}(e_2-e_1) = e_2^c-e_1^c+ \sum_{\Omega_2\backslash\Omega_1} \min(d_i) - \sum_{\Omega_1\backslash\Omega_2} \max(d_i). \qquad (4)$$

3.2 Computing scheduling risk

For unbounded distributions such as Gaussians, or even bounded distributions on wide intervals, we would expect (2) to almost never hold for non-trivial requirements. However, as was previously done in the context of PSTN's, one might be able to "restore" strong controllability to a PSTNU by "squeezing" probabilistic durations, therefore incurring non-zero amounts of scheduling risk (Definition 7).

Definition 7. *(Scheduling risk for strong policies) For a PSTNU N, let $C \subseteq \mathcal{C}_r$. Also, let $[l_i, u_i]$ be an externally-imposed bounding interval for the i-th contingent duration d_i in $\mathcal{C}_u$, so that (2) holds for every requirement constraint in C for a particular strong policy π_s. We define the scheduling risk $SR(\pi_s, C)$ of π_s with respect to C as*

$$SR(\pi_s, C) = 1 - \Pr\left(\bigwedge_{i=1}^{|\mathcal{C}_u|} d_i \in [l_i, u_i]\right). \qquad (5)$$

The joint distribution in (5) can be difficult to compute, or even completely unknown. In the next section, we present a scheduling algorithm that can not only minimize a guaranteed upper bound on (5), but also optimize other objectives while ensuring $SR(\pi_s, C) \leq \theta$ for a given $\theta \in [0, 1]$, the latter bound being referred to as a *chance constraint*.

4 Polynomial-time, risk-aware scheduling

We now introduce the Polynomial-time Algorithm for RIsk-aware Scheduling (PARIS), an algorithm leveraging a *linear program* (LP) formulation to extract (or determine the nonexistence of) risk-sensitive strong scheduling policies for PSTNU's. Different from the nonlinear approaches of (Tsamardinos 2002; Fang, Yu, and Williams 2014; Wang and Williams 2015), PARIS' linear formulation allows it to achieve not only dramatic speedups compared to the state of the art in risk-sensitive strong scheduling, but also, to the best of the authors' knowledge, be the first scheduler guaranteed to run in polynomial time.

4.1 Assumptions and walk-through

PARIS assumes that all events $e \in \mathcal{E}_u$ in a PSTNU are the endpoints of exactly one element of $\mathcal{C}_u$. If an event $e' \in \mathcal{E}_u$ is not associated with an uncontrollable duration, we make this event the endpoint of an STCU $[0, \infty)$ starting at the time reference ($t = 0$). Also, if $e' \in \mathcal{E}_u$ is the endpoint of two distinct elements of $\mathcal{C}_u$, we deem this a modeling error. This is because contingent constraints correspond to random continuous durations, and having them share an endpoint is equivalent to forcing random durations to be consistent with each other. Loops involving contingent durations are another modeling error: contingent durations are constrained to be positive, so a loop represents the inconsistent case of an event happening before itself.

The pseudo-code for PARIS is shown in Algorithm 1. For each requirement constraint in the PSTNU, line 5 uses Algorithm 2 to extract the set of events involved in (3) (contingent loops are detected in line 5 of Algorithm 2). The maximums and minimums in (4) are computed in line 6 by Algorithm 3, which are used in line 7 to enforce the necessary and sufficient conditions for strong controllability in (2). The functions `CtgLb` and `CtgUb` in Algorithm 3 return, respectively, externally-imposed lower and upper bounds for contingent constraints in terms of "squeezing variables", which are also used by function `SqueezeRisk` in Algorithm 1 (line 9) to compute a linear upper bound on (5). Line 11 of Algorithm 1 solves an LP to determine a risk-bounded strong scheduling policy π_s, and both π_s and an upper bound Λ_{SR} for (5) are returned. One should notice that Algorithm 2 will always return sequences ending in a controllable event (the e_i^c terms in (3)), which are returned by `Last` in lines 7 and 8 of Algorithm 3. Details on how these quantities are computed are given next.

4.2 A linear scheduling risk bound

Here we present the risk bound used as the objective in Algorithm 1 (line 9), allowing us to determine, from all available options, the strong scheduling policy π_s *minimizing this bound*. Section 4.5 shows how PARIS can be extended to handle other types of linear objectives as well.

In order to obtain a linear risk objective, we apply Boole's inequality to (5) and obtain the upper bound

$$\Lambda_{SR}(\pi_s, C) = \sum_{i=1}^{|\mathcal{C}_u|} \Phi_i(l_i) + (1-\Phi_i(u_i)) \geq SR(\pi_s, C), \qquad (6)$$

where Φ_i is the cumulative density function (cdf) associated with d_i. Notice that (6) holds regardless of whether contingent durations are independent or not, since it is obtained from Boole's inequality. As it stands, (6) is a combination

Algorithm 1: THE PARIS ALGORITHM.

```
Input: PSTNU $N$
Output: Strong policy $\pi_s$ and scheduling risk bound $\Lambda_{SR}$.
Function PARIS ($N$)
    $Obj \leftarrow 0$, $Cts \leftarrow \varnothing$
    for $c_r \in \mathcal{C}_r$ do
        $e_1, e_2 \leftarrow$ StartOf($c_r$), EndOf($c_r$)
        $d_1, d_2 \leftarrow$ CtgPath($N, e_1$), CtgPath($N, e_2$)
        $df_{mn}, df_{mx} \leftarrow$ DiffMinMax($N, d_2, d_1$)
        $Cts \leftarrow Cts \cup (df_{mn} \geq$ Lb($c_r$), $df_{mx} \leq$ Ub($c_r$))
    for $c_u \in \mathcal{C}_u$ do
        $Obj \leftarrow Obj +$ SqueezeRisk($c_u$)
        $Cts \leftarrow Cts \cup$ SqueezeCtrs($c_u$)
    $sol \leftarrow$ Minimize $Obj$ s.t. $Cts$
    $\pi_s \leftarrow$ ControllableEventValues($sol$)
    $\Lambda_{SR} \leftarrow$ Objective($sol$)
    return $\pi_s, \Lambda_{SR}$
```

Algorithm 2: EVENT'S CONTINGENT PATH.

```
Input: PSTNU $N$, event $e$.
Output: List of duration variables $dv$.
Function CtgPath($N, e$)
    $dv \leftarrow \varnothing$
    while $e \in \mathcal{E}_u$ do
        $c_u \leftarrow$ CtgDurationByEnd($N, e$)
        if $c_u \in dv$ then
            return ERROR
        else
            $dv, e \leftarrow dv \cup c_u$, StartOf($c_u$)
    return $dv \cup e$
```

of potentially nonlinear functions. Hence, we now develop efficient linear approximations of $\Phi(l_i)$ and $1 - \Phi(u_i)$ for several common models of contingent durations.

4.3 The risk of "squeezing" contingent durations

The $[l_i, u_i]$ bounds in (6) are externally imposed on contingent durations to cause (2) (strong controllability) to hold. If $[l, u]$ are the true bounds of a contingent duration, we see that imposing $l_i > l$ (squeeze lower bound) will cause $\Phi(l_i) > 0$ in (6). Analogously, choosing $u_i < u$ yields $(1 - \Phi(u_i)) > 0$. Therefore, *squeezing contingent duration bounds* causes (6) to grow. In the following, we show how we can quantify this risk as linear combinations of "squeezing variables" for common types of contingent durations.

Uniform durations Let $c_u \in \mathcal{C}_u$ be a PSTC representing a random uniform duration $d \sim U(l, u)$. Also, let s_l and s_u be, respectively, the amount by which one squeezes d's lower and upper bounds. In this case, we have

$$\begin{aligned}&\texttt{CtgLb}(c_u) = l + s_l, \texttt{CtgUb}(c_u) = u - s_u,\\ &\texttt{SqueezeRisk}(c_u) = \frac{s_l}{u-l} + \frac{s_u}{u-l},\\ &\texttt{SqueezeCtrs}(c_u) = s_l, s_u \in [0, u{-}l],\ \ s_l{+}s_u \leq u{-}l. \quad (7)\end{aligned}$$

It should be clear that $\Phi(l_i)$ and $(1{-}\Phi(u_i))$ in (6) correspond, respectively, to the first and second terms of SqueezeRisk(c_u) in (7). Fortunately, these are already linear functions of the squeeze variables for uniform distributions, so no approximations are required.

Algorithm 3: MINIMUM AND MAXIMUM IN (2).

```
Input: PSTNU $N$, duration variables $dv_1$ and $dv_2$ as in (3).
Output: Minimum and maximum in (2)
Function DiffMinMax($N, dv_2, dv_1$)
    $mx_1 \leftarrow mx_2 \leftarrow mn_1 \leftarrow mn_2 \leftarrow 0$
    for $i = 0, 1$ do
        for $(c_u \in dv_{(i+1)})\&\&(c_u \notin dv_{(2-i)})$ do
            $mn_{(i+1)} \leftarrow mn_{(i+1)} +$ CtgLb($c_u$)
            $mx_{(i+1)} \leftarrow mx_{(i+1)} +$ CtgUb($c_u$)
    $df_{mn} \leftarrow$ Last($dv_2$) $-$ Last($dv_1$) $+ mn_2 - mx_1$
    $df_{mx} \leftarrow$ Last($dv_2$) $-$ Last($dv_1$) $+ mx_2 - mn_1$
    return $df_{mn}, df_{mx}$
```

Set-bounded durations Since PSTN's were first introduced, STCU's have sometimes been treated as a particular type of PSTC, such as in (Tsamardinos 2002). Unfortunately, with respect to scheduling risk, the latter might not be true. In order to see that, let $c_u \in \mathcal{C}_u$ be an STCU with bounds $[l, u]$. For any nonzero amount of squeezing s_l of its lower bound, the true probability distribution of c_u could concentrate all probability mass in the interval $[l, l{+}s/2]$, and an analogous statement holds for the upper bound. This, in turn, would cause (6) to become a trivial bound $\Lambda_{SR} \geq 1$. Therefore, if (6) is to be a guaranteed non-trivial scheduling risk upper bound, *one must constrain STCU squeezing to be zero*, yielding the risk model

$$\begin{aligned}&\texttt{CtgLb}(c_u) = l, \texttt{CtgUb}(c_u) = u,\\ &\texttt{SqueezeRisk}(c_u){=}0, \texttt{SqueezeCtrs}(c_u){=}\varnothing. \quad (8)\end{aligned}$$

When $\mathcal{C}_u$ in a PSTNU only contains STCU's, (8) turns PARIS into a strong controllability checker for STNU's extending that of Vidal (1999) to the case where start events of contingent durations are not necessarily controllable.

Gaussians and other unimodal contingent durations A key distinction between PARIS and previous methods lies in its linear handling of common *unimodal* distributions for contingent durations, Gaussians being arguably the most common example. Instead of resorting to a nonlinear solver, here we develop *piecewise-linear upper bounds* (see Figure 3) for the risk terms in (6) involving Φ. Moreover, we exploit the fact that the pdf's of such distributions are monotonic on either side of the mode to derive piecewise-linear approximations of the cdf *without integer variables*, therefore yielding a *purely linear* formulation.

Let $f(x)$ be the pdf of a unimodal distribution, and let p_0 be its mode. Also, let $\mathbf{p}{=}\{p_{-m}, \ldots, p_{-1}, p_0, p_1, \ldots, p_n\}$ be a partition around p_0 with m segments to the left of the mode and n segments to the right (see Figure 3). For $l_i \in [p_j, p_{j+1}]$, $-m \leq j \leq -1$, one can write

$$\begin{aligned}\Phi(l_i) {\leq} &\left(\Phi(p_{-m}) {+} \sum_{k=-m}^{j-1} f(p_{k+1})(p_{k+1} {-} p_k)\right)\\ &+ f(p_{j+1})(l_i - p_j) \quad (9)\end{aligned}$$

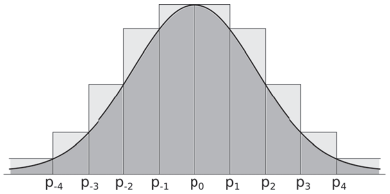


Figure 3: Piecewise-constant approximation of a Gaussian pdf used in the piecewise-linear upper bounds of $\Phi(l_i)$ and $(1-\Phi(u_i))$ in (6), where p_i's are given partition points.

Similarly, for $u_i \in [p_{j-1}, p_j]$, $1 \leq j \leq n$, we have

$$1-\Phi(u_i) \leq \left(1-\Phi(p_n)+\sum_{k=j}^{n-1} f(p_k)(p_{k+1}-p_k)\right) + f(p_j)(p_j-u_i). \quad (10)$$

The terms within parentheses in (9)-(10) are constants, as are all p_i's. Hence, these are, respectively, *piecewise-linear* upper bounds for $\Phi(l_i)$ and $1-\Phi(u_i)$. Seeking to incorporate (9)-(10) into (6), let $s_i \in [0, p_{i+1}-p_i]$ be the amount of squeezing in the interval $[p_i, p_{i+1}]$, $-m \leq i \leq n-1$. The squeezing risk model for unimodal distributions is given by

$$\begin{aligned}
&\texttt{CtgLb}(c_u)=p_{-m}+\sum_{i=-m}^{-1} s_i,\ \texttt{CtgUb}(c_u)=p_n-\sum_{i=0}^{n-1} s_i,\\
&\texttt{SqueezeRisk}(c_u)=\Phi(p_{-m})+(1-\Phi(p_n))\\
&\qquad\qquad +\sum_{i=-m}^{-1} f(p_{i+1})s_i+\sum_{j=0}^{n-1} f(p_j)s_i,\\
&\texttt{SqueezeCtrs}(c_u)=s_i \in [0, p_{i+1}-p_i], -m \leq i \leq n-1, \quad (11)
\end{aligned}$$

The s_i in (11) must consistently implement the piecewise-linear behavior from (9)-(10), i.e., for $i<0$, one must have $s_{i-1}<p_i-p_{i-1} \Rightarrow s_i=0$; and, for $i>0$, $s_i<p_{i+1}-p_i \Rightarrow s_{i-1}=0$. In general, these rules would have to be enforced through binary variables representing the "activation" of piecewise-linear segments. However, since I) s_i's in (11) only affect duration bounds through their sum; II) the coefficients of the s_i in `SqueezeRisk` are *monotonic* on either side of the mode; and III) PARIS minimizes (6), an optimal solution found by Algorithm 1 must necessarily fulfill the aforementioned rules, therefore correctly "squeezing" the distribution. Hence, *no binary variables are needed*!

4.4 Improving piecewise approximations

The previous section does not discuss how the partition $\mathbf{p}$ should be chosen. In principle, the specific p_i's should not matter, as long as the "box" between p_{i-1} and p_i is chosen so that it always overestimates the pdf (and, thus, the area under the curve). However, Figure 3 shows that such series of boxes can yield a rather crude approximation of the cdf, depending on where partition points are placed. Therefore, given a fixed number of partitions, we now focus on choosing $\mathbf{p}$ so that it approximates the cdf well. For that, let

$$g(\mathbf{p})=\sum_{i=-m}^{-1}(p_{i+1}-p_i)f(p_{i+1})+\sum_{j=0}^{n-1}(p_{j+1}-p_j)f(p_j) \quad (12)$$

be the total area of the piecewise approximation. Since (12) is an upper bound, one might seek to

$$\underset{\mathbf{p}}{\text{minimize}}\ g(\mathbf{p}) \text{ subject to } p_{i+1} \geq p_i, \forall i, p_{-m}, p_0, p_n \text{ fixed.} \quad (13)$$

The objective in (13) is not linear or convex in general (e.g., it is neither for Gaussians). Due to the inequalities, one could resort to SUMT (Fiacco and McCormick 1964), a.k.a. *barrier methods*, to solve a sequence of unconstrained minimization problems to provide increasingly better estimates of a local solution to (13). However, these intermediate steps are costly and, as pointed out by Boyd and Vandenberghe (2004), computing them exactly is not necessary. Therefore, seeking to keep computational requirements manageable, we resort to a simpler approach based on gradient descent. Starting with $\mathbf{p}^0=\mathbf{p}_0$, we compute

$$\mathbf{p}^{t+1}=Q\left(\mathbf{p}^t-\mu\nabla g\Big|_{\mathbf{p}=\mathbf{p}^t}\right) \quad (14)$$

until $\|\nabla g(\mathbf{p}^t)\| \approx 0$ or a maximum number of iterations is reached. In (14), the components of ∇g are

$$\begin{aligned}
\frac{\partial g}{\partial p_k}&=f'(p_k)(p_k-p_{k-1})+f(p_k)-f(p_{k+1}), -m<k<0,\\
\frac{\partial g}{\partial p_k}&=f'(p_k)(p_{k+1}-p_k)+f(p_{k-1})-f(p_k), 0 \leq k<n, \quad (15)
\end{aligned}$$

and 0 for $k \in \{-m, 0, n\}$; μ is a positive constant; and $Q(\cdot)$ is a projection ensuring (14) remains in the feasible region of (13). In our implementation, $Q(\cdot)$ is the identity if its argument is feasible, and otherwise outputs a random feasible perturbation around $\mathbf{p}^t$. Thus, at any t, (14) is feasible and, for small enough μ, will improve the upper bound (12) at every iteration. For a Gaussian $N(\mu, \sigma^2)$, we have

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)}, f'(x)=f(x)\left(\frac{\mu-x}{\sigma^2}\right). \quad (16)$$

4.5 From minimum risk to other linear objectives

The description of PARIS in Algorithm 1 computes π_s minimizing the scheduling risk bound (6). However, as pointed out in (Fang, Yu, and Williams 2014; Wang and Williams 2015), there might be situations where other types of linear objectives (e.g., the schedule's makespan) might be preferred over minimizing risk. Thus, let

$$\underset{\mathbf{x}}{\text{minimize}}\ h(\mathbf{x}) \quad \text{subject to } Cts(\mathbf{x}) \quad (17)$$

be the *target problem*, where $h(\mathbf{x})$ is the *desired objective* (e.g., the makespan) for PARIS; $\mathbf{x}$ is the vector of problem variables (schedule of controllable events, squeezing variables, etc.); and Cts are the same as in line 11 of Algorithm 1, plus any additional (linear) ones required by $h(\mathbf{x})$. In this section, we analyze two risk-motivated settings for solving (17): I) a *chance-constrained* setting, in which we solve (17) without caring for the actual value of the scheduling risk $SR(\mathbf{x})$, as long as one can ensure that it is bounded by a user-specified tolerable risk level θ; and II) a *tight risk gap* setting, in which we would like to solve (17) while ensuring that our estimate of $SR(\mathbf{x})$ is as good as possible.

Chance-constrained In the *chance-constrained* setting, (17) must be solved while ensuring $SR(\mathbf{x}) \leq \theta$ for a given $\theta \in [0, 1]$. Following Section 4.2, we achieve $SR(\mathbf{x}) \leq \theta$ through the sufficient condition

$$\Lambda_{SR}(\mathbf{x}) \leq \theta, \tag{18}$$

which should be added as an element of $Cts(\mathbf{x})$ in (17). It is worthwhile to notice that (18) is sufficient to enforce $SR(\mathbf{x}) \leq \theta$ *even in the absence of binary values* to ensure that the "squeezings" in (11) are performed correctly! The reasons are the same as presented in Section 4.3: squeeze variables have monotonic coefficients and can only affect bounds through their sums, so a bound $\Lambda'_{SR}(\mathbf{x})$ resulting from "potentially incorrectly squeezed" distributions can only overestimate the bound $\Lambda_{SR}(\mathbf{x})$ generated by the same amount of squeezing, but performed in the correct order. Therefore, since $\Lambda'_{SR}(\mathbf{x}) \leq \theta$ is sufficient for (18), a solution of (17) is guaranteed to be chance constrained *no matter how the squeezings are performed.*

Tight risk gap Assume now that, in addition to solving (17), we require the scheduling risk bound given by (6) to be tight. This is useful, for instance, when no knowledge about the joint distribution of durations is available, so that (5) cannot be computed even if all bounds $[l_i, u_i]$ for the contingent durations are given. In this case, (6) becomes our best estimate of the scheduling risk for the mission, and we would like to make it as good as possible.

A natural way of achieving this goal would be to *introduce binary variables* in (17), so that the piecewise-linear approximations in Section 4.3 are correctly implemented. However, that would turn (17) into a significantly harder problem to solve, since it would become a Mixed-Integer LP (MILP). Therefore, we propose to replace (17) by the surrogate

$$\underset{\mathbf{x}}{\text{minimize}} \ \Lambda_{SR}(\mathbf{x}) + Mh(\mathbf{x}) \quad \text{subject to } Cts(\mathbf{x}), \tag{19}$$

where $\Lambda_{SR}(\mathbf{x})$ is the risk bound (6) and M is a finite positive constant. Unlike (17), (19) *does not require binary variables* due to the introduction of $\Lambda_{SR}(\mathbf{x})$ in the objective. On the other hand, (19) no longer optimizes our desired objective $h(\mathbf{x})$. Despite the latter, we now show that M can be chosen so that a solution to (19) approximates that of (17) with arbitrary precision while requiring no integer variables.

Let $\mathbf{x}^*$ be optimal solution found by (17), and let $h^* = h(\mathbf{x}^*)$ be the corresponding minimal value of $h(\cdot)$ over the convex region defined by $Cts(\mathbf{x})$. Also, let $\mathbf{x}'$ be the solution found by (19) over the same convex region, with corresponding desired objective $h(\mathbf{x}') = h^* + \Delta h$. The Δh term is the *objective degradation*, and must be such that $\Delta h \geq 0$, given that h^* is the minimum of $h(\cdot)$ over $Cts(\mathbf{x})$. Any such degradation in $h(\mathbf{x})$ must be a result of a corresponding decrease $\Delta\Lambda_{SR}(\mathbf{x})$ of the risk bound, and the minimality of (19) implies $\Delta\Lambda_{SR}(\mathbf{x}) + M\Delta h \leq 0$. The last step is noticing that the risk bound improvement is such that $-\Delta\Lambda_{SR}(\mathbf{x}) \leq |\mathcal{C}_u|$, which is the difference between the maximum ($\Lambda_{SR} = |\mathcal{C}_u|$) and minimum ($\Lambda_{SR} = 0$) values of Λ_{SR} in (6). Therefore, we arrive at the degradation bound

$$\Delta h \leq \frac{|\mathcal{C}_u|}{M}. \tag{20}$$

For instance, if $h(\mathbf{x})$ is the schedule's makespan measured in seconds and one chooses $M = 1000|\mathcal{C}_u|$, (20) guarantees that the *fully linear* surrogate (19) approximates the true minimal makespan h^* with millisecond precision! Also, nothing prevents us from imposing (18) on (19) to enforce $SR(\mathbf{x}) \leq \theta$. In Section 5, this will be referred to as the *tight, chance-constrained* (TCC) setting.

4.6 Algorithm properties

This section presents soundness, completeness, and complexity properties for the different formulations of PARIS.

Lemma 1. *PARIS is sound.*

Proof: PARIS enforces (4), which are necessary and sufficient for strong controllability. Therefore, any schedule found must be a strong scheduling policy. □

Lemma 2. *PARIS runs in polynomial time.*

Proof: the loops in lines 3 and 8 of Algorithm 1 run a polynomial number of iterations, and both Algorithms 2 and 3 run in polynomial time. Also, the scheduling risk models in (7), (8), and (11) create a polynomially-large number of variables and constraints relative to the number of contingent durations, and Karmarkar (1984) showed that the LP in line 11 of Algorithm 1 can be solved in polynomial time. Finally, the cap on the number of iterations for the partition optimization step (14) introduces a maximum overhead that grows linearly with the number of contingent durations. □

Lemma 3. *When contingent durations are restricted to STCU's or uniform PSTC's, PARIS is complete.*

Proof: the squeezing models (7) and (8) consider the whole range of possible externally-imposed upper and lower bounds for contingent durations. Therefore, if there are squeezings for which a strong policy exists, PARIS will find it. Otherwise, it will return no solution. □

Lemma 4. *The squeezing model (11) for unimodal PSTC's renders PARIS incomplete.*

Proof: the model (11) will fail to return strong schedules requiring upper and lower bounds to be squeezed beyond the extrema of the interval, or not containing the mode. □

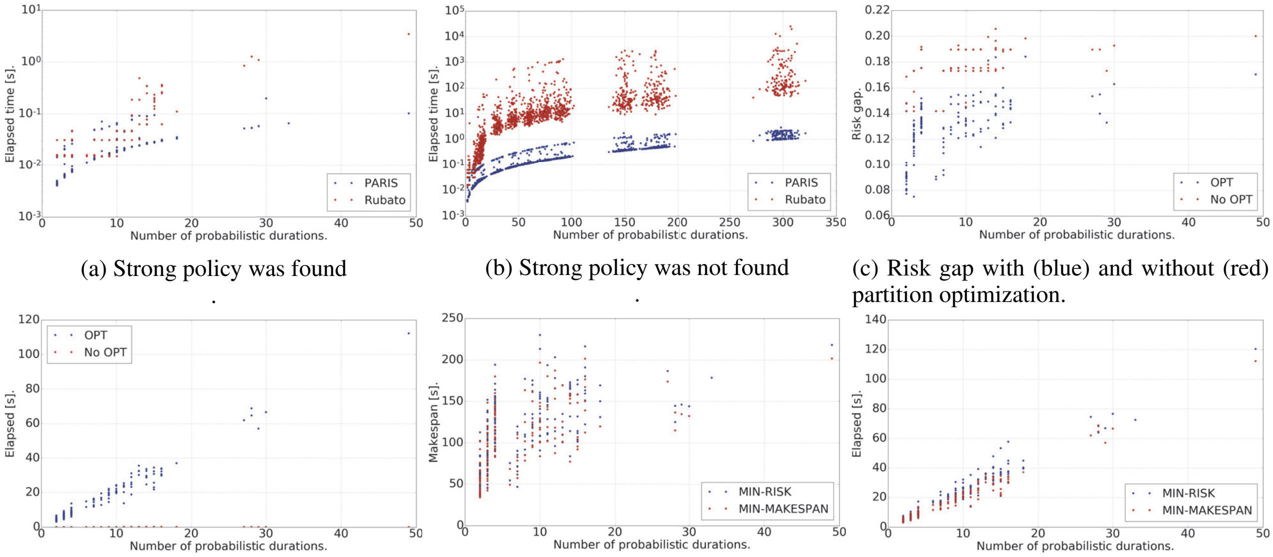
Lemma 5. *Chance-constrained PARIS is sound, but incomplete.*

Proof: Soundness of (18) follows from $SR(\mathbf{x}) \leq \Lambda_{SR}(\mathbf{x}) \leq \theta$. Incompleteness follows from the fact that enforcing (18) to ensure $SR(\mathbf{x}) \leq \theta$ may discard valid solutions when θ is placed in the gap between the minimum of $\Lambda_{SR}(\mathbf{x})$ and the true value of $SR(\mathbf{x})$. □

5 Experiments

The empirical evaluation of PARIS leveraged two distinct datasets: *CAR-SHARING* and *ROVERS*, both available at http://mers.csail.mit.edu/datasets/scheduling, along with several other examples.

The CAR-SHARING dataset, first made available by Fang, Yu, and Williams (2014) and later used by Wang and Williams (2015), served the purpose of evaluating PARIS against the state of the art in PSTN scheduling. Experiments



(a) Strong policy was found

(b) Strong policy was not found

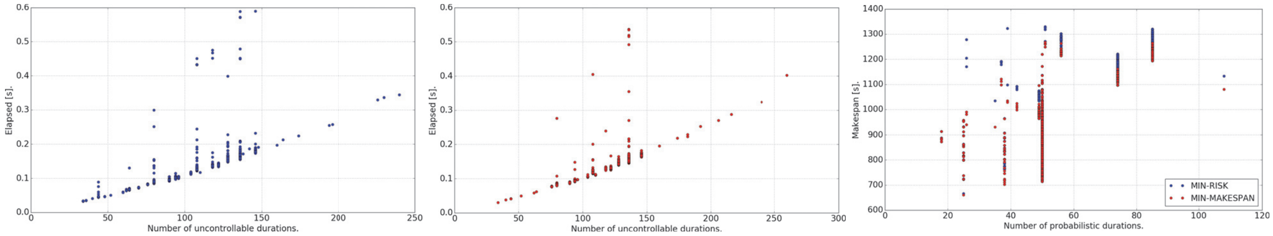
(c) Risk gap with (blue) and without (red) partition optimization.

(d) Elapsed time for 4c.

(e) Min. risk (blue) and min. makespan.

(f) Elapsed time for 4e.

Figure 4: Performance of Rubato and PARIS on CAR-SHARING dataset.



(a) Strong policy was found.

(b) Strong policy was not found.

(c) Min. risk (blue) and min. makespan.

Figure 5: Performance of PARIS on ROVERS dataset.

by Wang and Williams (2015) on this dataset show that *Rubato*, their SMT-based chance-constrained strong controllability checker for PSTN's, outperforms Tsamardinos's (2002) approach and the *Picard* system presented in (Fang, Yu, and Williams 2014) by nearly an order of magnitude. Therefore, here we compare Rubato's performance relative to that of PARIS on CAR-SHARING. Rubato is implemented in Common Lisp and uses Ipopt (Wächter and Biegler 2006) as the nonlinear solver, while PARIS is implemented in Python and uses Gurobi 6.0.4.

The ROVERS dataset consists of 4380 randomly-generated PSTNU instances modeling planetary rover coordination scenarios similar to the one depicted in Figure 1a. ROVERS instances feature the coordination between two to ten rovers, which have to complete between one to ten sequential exploration tasks (*drive*, *drill*, *collect*, and *process*) in parallel before reconvening at a relay location and transmitting their data to a satellite within a given visibility window. ROVERS contains all combinations of uncontrollable temporal durations discussed in this work and, to the best of the author's knowledge, cannot be handled by existing scheduling algorithms.

Our first test consisted of running the risk-bound minimization version of PARIS explained in Algorithm 1 against Rubato on CAR-SHARING, with results shown in Figures 4a and 4b (vertical axes in log scale). Out of 1800 instances, PARIS found strong policies for 186 in Figure 4a (same number as in (Fang, Yu, and Williams 2014)), while Rubato found 142. Unsurprisingly, we see that strong schedules tend to exist for networks with fewer uncontrollable durations. PARIS handled Gaussian durations using 8 equally-spaced partitions (length equal to σ) on either side of the mean. Notice that Figures 4a and 4b shows PARIS outperforming Rubato by about 1 or 2 orders of magnitude in most test instances, and up to three or four orders in some of the most difficult problems. Also, while PARIS computes the strong policy, Rubato solves the strictly easier problem of checking if one such policy exists.

For the 186 instances for which PARIS found a strong policy, our next test on CAR-SHARING evaluated how the partition optimization procedure from Section 4.4 impacts the gap between the risk bound in (6) and the true scheduling risk in (5). Since (5) is hard to compute in general, we assumed, for this particular experiment only, that all contingent durations were independent, in which case (5) becomes

$$SR(\pi_s, C) = 1 - \prod_{i=1}^{|\mathcal{C}_u|} \Pr(d_i \in [l_i, u_i]). \tag{21}$$

Gradient descent was allowed a maximum of 12000 iterations with gradient norm tolerance 10^{-3} and fixed $\mu = 0.03$. Figure 4d shows the elapsed time as a function of the number of uncontrollable duration, while Figure 4c shows the risk gap between (6) and (21) for different instances with and without partition optimization. Partition optimization caused the risk gap to improve on all instances, with an average gap improvement of 4.8% (absolute value, not relative). Also, the linear trend in Figure 4d confirms that partition optimization *does not affect* PARIS' polynomial-time complexity.

Our last test on CAR-SHARING evaluated the effectiveness of PARIS in optimizing general linear objectives, as explained in Section 4.5. For this test, the *schedule makespan* was used as the desired objective in (19), with a chance constraint θ=30% imposed through (18). It was compared against the risk-bound minimization version in Algorithm 1, and the results are shown in Figures 4e and 4f. As expected, using (19) with makespan as the desired objective improved the makespan for all test instances, with an average reduction of 8.5 seconds. Also, Figure 4f shows that both formulations have similar runtimes.

On the ROVERS dataset, from a total of 4380 PSTNU instances, PARIS found strong policies for 2840 of them. From those, 911 had probabilistic durations squeezed to a single value, i.e., even though a strong policy exists, it is almost guaranteed to fail. Different from CAR-SHARING, Figures 5a and 5b show that there are instances of ROVERS featuring both strong policies and a large number of uncontrollable durations. Moreover, we observe from these figures the small amount of time required by PARIS to solve instances with and without strong policies, reinforcing the claim that PARIS is suitable for hardware with computational and energy constraints. Finally, the same conclusions from CAR-SHARING regarding partition optimization and makespan optimization hold on ROVERS. For instance, as shown in Figure 5c, makespan optimization with a chance constraint θ=20% improved the makespan for all problem instances, with an average reduction of 37.36 seconds.

6 Conclusions

This work introduces PSTNU's, a temporal modeling formalism subsuming STNU's and PSTN's that captures different levels of knowledge about sources of temporal uncertainty. It also presents PARIS, a provably polynomial-time and sound algorithm for risk-sensitive strong scheduling of PSTNU's that outperforms the current fastest algorithm for PSTN strong scheduling by several orders of magnitude. Due to their significantly reduced computational requirements, we hope that our results will serve to endow temporal planners in general, and autonomous robotic agents in particular, with a keen sensitivity to scheduling risk.

Acknowledgements

This research was partially funded by the AFOSR grant FA95501210348. The second author is funded by the Keck Institute for Space Studies (KISS). We would like to thank Peng Yu for making the CAR-SHARING dataset available, and Christian Muise and our anonymous reviewers for their constructive and helpful comments.

References

Beaudry, E.; Kabanza, F.; and Michaud, F. 2010. Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. In *ICAPS*, 10–17.

Birge, J. R., and Louveaux, F. V. 1997. *Introduction to stochastic programming*. Springer.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

Cimatti, A.; Micheli, A.; and Roveri, M. 2015. Strong temporal planning with uncontrollable durations: a state-space approach. AAAI.

Coffrin, C., and Van Hentenryck, P. 2014. A linear-programming approximation of AC power flows. *INFORMS Journal on Computing* 26(4):718–734.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 1–96.

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

Fang, C.; Yu, P.; and Williams, B. C. 2014. Chance-constrained probabilistic simple temporal problems. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Fiacco, A. V., and McCormick, G. P. 1964. The sequential unconstrained minimization technique for nonlinear programing, a primal-dual method. *Management Science* 10(2):360–366.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, 302–311. ACM.

Micheli, A.; Do, M.; and Smith, D. E. 2015. Compiling away uncertainty in strong temporal planning with uncontrollable durations. In *International Joint Conference on Artificial Intelligence*.

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

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

Wächter, A., and Biegler, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106(1):25–57.

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