Robustness Computation of Dynamic Controllability in Probabilistic Temporal Networks with Ordinary Distributions

Michael Saint-Guillain[1*] , **Tiago Stegun Vaquero**[2] , **Jagriti Agrawal**[2] and **Steve Chien**[2]
[1]Université Catholique de Louvain (UCLouvain), Belgium
[2]Jet Propulsion Laboratory, California Institute Technology, CA, USA
michael.saint@uclouvain.be, {tiago.stegun.vaquero, jagriti.agrawal, steve.chien}@jpl.nasa.gov

Abstract

Most existing works in Probabilistic Simple Temporal Networks (PSTNs) base their frameworks on well-defined probability distributions. This paper addresses on PSTN Dynamic Controllability (DC) robustness measure, *i.e.* the execution success probability of a network under dynamic control. We consider PSTNs where the probability distributions of the contingent edges are ordinary distributed (*e.g.* non-parametric, non-symmetric). We introduce the concepts of dispatching protocol (DP) as well as DP-robustness, the probability of success under a predefined dynamic policy. We propose a fixed-parameter pseudo-polynomial time algorithm to compute the exact DP-robustness of any PSTN under *NextFirst* protocol, and apply to various PSTN datasets, including the real case of planetary exploration in the context of the *Mars 2020* rover, and propose an original structural analysis.

1 Introduction

Temporal networks formalize the arrangement and interdependencies of tasks, or activities, that compose an operational project. In a simple temporal network (STN), activities are modeled as a finite set of time events. In practice, some activity durations, considered as *contingent*, remains unknown beforehand. In the case some stochastic knowledge on the uncertain durations exists, then one can model it as (estimated) probability distributions, leading to the extending concept of probabilistic STN, or *PSTN*. Solving a STN then amounts at finding an assignment of time values to events that fulfils all the constraints between events. Whenever such schedule exists, a network is said to be *controllable*. When the operational assumptions enable it, the schedule may be *dynamically constructed*, the time values being assigned as durations are observed. Yet, even under dynamic decision, due to unfortunate durations a network may reveal uncontrollable. How likely is a PSTN to lead to a successful execution?

Provided some stochastic knowledge on contingent activity durations, the degree of dynamic controllability (DDC) of a PSTN can be quantified, as the success probability of a given task network. The current literature proposes DDC computation methods for PSTNs that involve *unimodal distributions only* (uniform or normal), or exploit Monte Carlo simulation, which admits ordinary distributions. On both cases, the computed DDC values are *approximations* only, without guarantee on the true robustness of the network.

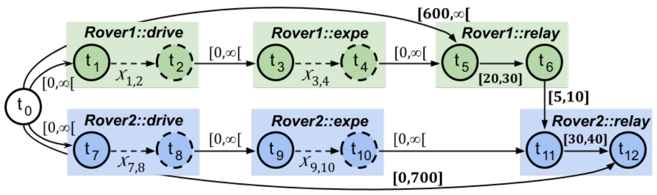

Figure 1: A simplified hypothetical sol on Mars for two planetary rovers, encoded as a PSTN. Bold: controllable. Dashed: contingent.

Contributions. We propose the first efficient DDC (*aka.* robustness) computation method capable of dealing with PSTNs with *any possible ordinary probability distributions* (*i.e.* non-parametric, non-symmetric, multi-modal, or even hand-made). Our method computes a *valid lower bound on the exact* DDC of a PSTN. Under a specific dispatching protocol, it computes the *exact* execution success probability. Furthermore, it enables to compute a lower bound on each task's own success probability within that network, which can be mapped to activity temporal brittleness. On the application side, we propose a new method for identifying structural bottlenecks in temporal brittleness analysis, applied to the real case study of the Mars 2020 rover's task networks.

2 Temporal Networks

STN. Simple Temporal Network is a popular formalism for temporal constraint reasoning [Dechter *et al.*, 1991], framed as a constraint satisfaction problem over time point variables: a STN is a tuple $\langle T, C \rangle$, where T is a set of time points ($t_i \in T \subseteq \mathbb{R}$) and C is a set of constraints $c(t_i, t_j)$ that encode bounds on the differences between pairs of time points: $l_{ij} \leq (t_j - t_i) \leq u_{ij}$, *i.e.* $(t_j - t_i) \in [l_{ij}, u_{ij}]$. A solution is called a *schedule*, a specific assignment to all $t_i \in T$. Constraint $c(t_i, t_j)$ represents the valid bounds for the duration of an activity defined by time points (t_i, t_j), whereas $t_j - t_i$ gives the activity duration set up by a specific schedule.

*Contact Author

Probabilistic STN. Most realistic operational contexts account for temporal uncertainty. **PSTN** is a natural extension of STN in which probability density functions are associated to activity durations [Tsamardinos, 2002]. The PSTN formalism partitions both sets T and C as $T = T_E \cup T_C$, where *executable* time points T_E are determined by the agent, and *contingent* time points T_C are assigned by nature. The constraints set is $C = C_R \cup C_C$ where: *requirement* edges, C_R, are controlled by the agent; and *contingent* edges, C_C, are determined by nature, each element being described as a probability distribution $(t_j - t_i) = X_{i,j}$. A schedule assigns values to executable time points only. The duration $(t_j - t_i)$ associated to any contingent time point t_j remains unknown prior to execution. In absence of stochastic knowledge for a contingent duration $(t_j - t_i)$, a common usage is to consider it uniform: $X_{ij} \sim U(l_{ij}, u_{ij})$. A temporal network in which that assumption applies to all its contingent edges is called a STN with Uncertainty: **STNU** [Vidal and Ghallab, 1996].

Illustrative example: rover operations. An hypothetical example of Mars rovers PSTN is depicted in Fig. 1. Each rover has three activities in sequence: drive towards a science site, experiment, and relay results to an orbiter. A special time point $t_0 = 0$ represents the beginning of the operations. Time events are linked by temporal constraints, either controllable or contingent. In our example, the rovers work independently during their driving and science activities. They do not coordinate until the communication time window, which strictly happens between time 600 to 700. Communication tasks cannot overlap, and *Rover1* is chosen to relay first. However, duration of driving and experimental activities are highly uncertain. In practice, distributions can be estimated from historical observations. Even an inaccurate stochastic knowledge, *e.g.* obtained accurate observations, leads to valuable results in practice (as illustrated in [Saint-Guillain, 2019] for Mars-inspired operations). In Fig. 1, distribution $X_{1,2}$ describes the stochastic duration of driving activity (t_1, t_2), encoded in the PSTN as a contingent constraint $c(t_1, t_2) \in C_C$, $t_2 \in T_C$.

3 Robustness

Ideally, a perfect assignment of all time points in T_E would work for any situation imposed by nature. In practice that is very restrictive, if not impossible. Instead, we refer to a *strategy* mapping observations to schedules during execution. In fact, if the PSTN of Fig. 1 involves some probability distribution with unbounded tail, then such perfect schedule does not exist. Under uncertain activity durations, how likely is the execution of a PSTN to succeed? What is the probability that our rovers get their relay activities ($\geq$20 long for *Rover1*, $\geq$30 long for *Rover2*) during the communication window?

3.1 Controllability

In STNU and PSTN realm, consistency is not directly applied due to the unpredictable assignment of contingent time points and edges. A PSTN relies instead on checking *controllability*, which verifies whether an agent can generate a consistent schedule to any situation that may arise in the external world. Controllability theory is usually applied to STNUs, but can also be applied to more general PSTNs (distributions being not restricted to uniform ones), at specific different levels.

An PSTN is said to be *strongly controllable* (SC) [Vidal and Ghallab, 1996] *iff* there exists at least one *strong schedule* (*a.k.a.* static schedule), *i.e.* a "universal" schedule that fits any situation, guaranteed to satisfy all temporal constraints regardless of the nature's assignments. That is motivated by cases where agents have to compute a schedule offline before making any observations, with no opportunity to adapt online. Nevertheless, in practice a valid static schedule is rarely available in dynamic and unpredictable environments.

A more practical level would be *dynamically controllable* (DC) [Morris *et al.*, 2001; Morris, 2014], in which we check whether there exists an execution strategy such that, at any time during execution, the partial sequence executed so far extends to a complete solution, whatever durations remain to be observed. It requires the agent to be able to determine, in a dynamic fashion, a valid assignment of executable time points based on observed past contingent ones, without violating any future temporal constraints.

3.2 Degrees of Dynamic Controllability

Whereas controllability checking has been proven polynomial in many cases [Bhargava and Williams, 2019], evaluating the degree of controllability of uncontrollable networks is still an open problem. In [Akmal *et al.*, 2019], the *degree of dynamic controllability* (DDC) of a STNU is introduced. When considering uniform distributions only, the DDC is defined as the proportion of contingent edges realizations in which the temporal network remains dynamically controllable.

In the context of a PSTN, the DDC must rather be defined in terms of the probability mass of the controllable realization space. That definition of DDC is equivalent to the robustness measure, independently introduced in [Brooks *et al.*, 2015], as the probability of the network to be successfully executed under dynamic control. Assuming discrete distributions:

$$DDC(N) \equiv \sum_{\xi \in \Omega} \mathbb{P}\{\xi\} \; \Phi(N, \xi) \qquad (1)$$

where Ω is the set of all possible realizations of the random contingent edges' duration. The deterministic function $\Phi(N, \xi)$ takes value 1 *iff* the network N is dynamic controllable in scenario (*a.k.a.* situation) ξ. In fact, in both cases and given scenario ξ, a PSTN reduces to a regular STN, as the contingent edges get assigned fixed durations. Whereas $\Phi(N, \xi)$ can be checked in linear time, the computation of (1) is intractable in practice, as the size of Ω grows exponentially with the number of contingent edges.

3.3 Dispatching Protocols

Operational contexts such as space missions usually forbid recomputing a schedule in the middle of the operations [Chi *et al.*, 2019]. Yet, the use of a static schedule is often either impossible in practice, or comes with a significant waste in terms of operational time. Such approach is currently operating *Curiosity* rover, with static schedules that overestimate processing times by 30% in average [Gaines *et al.*, 2016].

Future planetary rover M2020 will be equipped with a non-backtracking onboard scheduler, designed to take online decisions based on current observations [Agrawal *et al.*, 2019; Chi *et al.*, 2018; Rabideau and Benowitz, 2017]. However, because of computational limitations, such online decisions must remain very light, thus following a predefined *strategy*: a *dispatching protocol* (DP).

We derive the concept of DP from that of recourse strategy [Birge and Louveaux, 2011]. Given the state at time t and new observation ξ^t, a DP function returns an assignment of (a subset of) the n remaining executable time points:

$$T_E \rightarrow \mathbb{R}^n = \text{DP}(N, \xi^t)$$

We distinguish three families of DP's, depending on the complexity of $\text{DP}(\cdot)$. First, the case where $\text{DP}(\cdot)$ is constant time directly corresponds to a *static/strong schedule*. Second, the case of non-polynomial time functions generally stands for an *online reoptimization process*, aimed at solving the inherent multistage stochastic program. In that case, there is no predefined strategy, a new scheduling problem is solved at each time step, regardless the computational expense.

Finally, the case of a (non-constant) polynomial-time $\text{DP}(\cdot)$ allows agents to *dynamically adapt the schedule*, while being *computationally limited*. For example, Rabideau and Benowitz [2017] describe a $\mathcal{O}(n^2)$ quadratic $\text{DP}(\cdot)$ protocol to be computed by the future M2020 onboard scheduler, in order to make online decisions based on observations and pre-optimized parameters [Chi *et al.*, 2019]. Brooks *et al.* [2015] consider a linear time protocol, called *NextFirst* protocol, which we further describe below.

DP-Robustness. Most of existing work in the literature aims at computing the DDC in (1). Yet, one may be interested in the success probability of a network N under a specific dynamic dispatching protocol $\mathcal{P}$, the DP-robustness:

$$r^{\mathcal{P}}(N) = \sum_{\xi \in \Omega} \mathbb{P}\{\xi\} \; \Phi^{\mathcal{P}}(N, \xi) \tag{2}$$

where $\Phi^{\mathcal{P}}(N, \xi)$ returns 1 *iff* following protocol $\mathcal{P}$ in situation ξ leads to a successful execution. Given a dispatching protocol $\mathcal{P}$, the DP-robustness necessarily constitutes *a lower bound on the true DDC* of a network:

$$\forall \mathcal{P}: \quad r^{\mathcal{P}}(N) \leq DDC(N).$$

It is worth noting that in many operational contexts, under computationally limited settings (*e.g.* Mars 2020 rover), a DP-robustness measure based on an appropriate protocol may be more adequate than the DDC, as the latter relies on optimal online re-scheduling, which is intractable in general.

***NextFirst* dispatching protocol.** The *NextFirst* protocol [Brooks *et al.*, 2015], also known as *DC-dispatch* [Morris *et al.*, 2001], achieves a $\mathcal{O}(n)$ linear time dynamic control by starting activities as soon as possible. Let t_j be a controllable time point in a PSTN, and $I_j = \{(0, j), \ldots, (i, j)\}$ the set of incoming edges in t_j. We assume I_j to contain controllable edges only, which one can easily enforce as shown in Fig. 2(b). Therefore, t_j is assigned a time value as soon as all the preconditions are validated, that is, all the $t_0, \ldots, t_i$ time points are known:

$$t_j = \max(t_0 + l_{0j}, \; \ldots \; , t_i + l_{ij}). \tag{3}$$

In the case $t_j > \min(t_0 + u_{0j}, \; \ldots \; , t_i + u_{ij})$, the dynamic execution is considered as failed, and is interrupted. Back to our PSTN example in Fig. 1, the value of t_{11} is then dynamically set to $\max(t_{10}, \; t_6 + 5)$ as soon as tasks *Rover2:expe* and *Rover1:relay* are completed. Execution fails if t_{11} exceeds $t_6 + 10$. Eventually, we hope for $t_{13} \leq 700$.

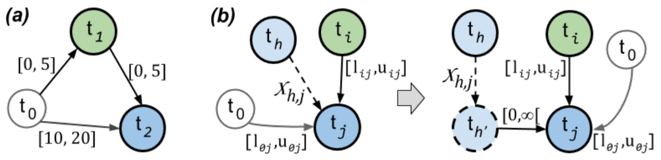

Figure 2: (a) An example of *NextFirst* incompleteness on a basic STN. (b) The network can be transformed in order to avoid synchronisation points involving contingent incoming edge(s).

	Comp. value		**Prob. distribution**			
	≤	≃	∼U	∼N	All	Cont.
Monte Carlo		✓	✓	✓	✓	✓
[Akmal *et al.*, 2019]		✓	✓			✓
[Cui *et al.*, 2015]		✓	✓			✓
[Vaquero *et al.*, 2019]		✓	✓	✓		✓
[Cesta et al., 1998]		✓	✓			
[Wilson *et al.*, 2014]		✓	✓			
[Saint-Guillain, 2019]		✓	✓	✓	✓	
Our proposal	✓	✓	✓	✓	✓	

Table 1: Existing DDC measurement methods. The key properties we consider are **1)** Quality of the *computed value*: lower bound (≤) on DDC, approximation (≃) of DDC, and **2)** Properties of the supported *probability distributions*: Uniform, Normal, any ordinary distribution, and finally whether distributions can be continuous).

Incompleteness of *NextFirst*. The simplicity of *NextFirst* enables efficient DP-robustness computation, but comes at the expense of being incomplete. A more clever dispatching protocol assigns a time value based not only on past time events, but also remaining ones. Fig. 2(a) shows an example of a basic (deterministic) STN, for which operating t_1 as soon as possible, following (3), leads to a failure.

4 Landscape of Robustness Measures

Table 1 summarizes the existing contributions on computing (or approximating) the DDC of a network. Recent studies, such as [Cui and Haslum, 2019], focus on computing valid dynamic decisions for dynamically controllable networks, whereas we focus on uncontrollable PSTNs. Finally, we do not cover conditional (P)STNs [Williams *et al.*, 2009], nor (P)STNs with resource usage [Kumar *et al.*, 2018].

In the context of dynamic controllability, the work of [Akmal *et al.*, 2019] is the first attempt to compute the true DDC of a network. They propose an approximation technique, achieving good accuracy rate. In [Brooks *et al.*, 2015], a Monte Carlo sampling approach approximates the DDC, under the *NextFirst* protocol. Other approximated robustness metrics have been considered: Cesta *et al.* [1998] and Wilson *et al.* [2014] see the robustness of a network as a potential of solution flexibility (*i.e.* aggregate time slack). They coarsely approximate how easily a schedule can be adapted during operations. A quite similar approach has been proposed by [Tsamardinos, 2002], by reasoning on the probability distributions describing the gaps between the time constraint bounds. All these approaches suffer from the fact that they do not account for correlation between temporal constraints, hence over-estimating the true robustness (or flexibility). Cui *et al.* [2015] define the robustness of STNUs (i.e. under non-probabilistic uncertainty) as the maximum variations that all the contingent durations may face while still having strong/dynamic control. Based on the same idea, Vaquero *et al.* [2019] define the notion of activity temporal brittleness, by analysing how much duration deviation (based on a distribution) each activity, taken separately, can absorb before the network becomes dynamically uncontrollable. [Saint-Guillain, 2019] constitutes the first attempt at computing a valid lower bound on the DDC, handling ordinary distributions. Yet, their computational method applies to a particular PSTN only, leading to an approximation in the general case. Although not expressed by Table 1, only sampling based methods, such as Monte Carlo, allow to consider PSTNs with dependent contingent constraints realizations.

5 Exact Computation of DP-robustness

Assumptions and network preprocessing. We assume discrete time horizon and probability distributions. The horizon is noted $H = 1..h$. We also assume independence between the activity duration probabilities. Let then $p^d_{ij} = \mathrm{P}(t_j - t_i = d)$ be the probability that the uncertain activity duration, represented by contingent edge (i, j), is of $d \in H$ time units. The dynamic execution follows the *NextFirst* dispatching protocol. To each time point $t_i \in T$ is always associated a constraint $[l_{0i}, u_{0i}]$, defining the valid time window w.r.t. t_0. If no such is specified, we assume $[0, \infty[$. Following Fig. 2(b) we transform the network to avoid contingent synchronisation points. We call $t_{\text{leaf}} \in T_C$ the *final time event* (*e.g.* t_{12} in Fig. 1). In case of multiple final events, we add a final synchronisation one, that links these with $[0, \infty[$ edges.

Exact computation. We now describe how to compute the DP-robustness $r^{\text{nf}}(N)$ of a network N, using closed-form expressions instead of (1), when using *NextFirst* protocol.

We are interested in the probability that every controllable time point gets assigned a time unit within its boundaries:

$$r^{\text{nf}}(N) = \mathbb{P}\Big\{ \bigwedge_{t_j \in T_c} t_j \le \min(t_0+u_{0j}, \dots, t_i+u_{ij}) \Big\} \quad (4)$$

$$= \mathbb{P}\Big\{ t_{\text{leaf}} \le \min(t_0+u_{0j}, \dots, t_i+u_{ij}) \Big\} = \sum_{t \in H} P_{\text{leaf}}(t) \quad (5)$$

A little abuse of notations: probability $\mathbb{P}\{\alpha = \top\}$ for a logical formula α to be true is denoted $\mathbb{P}\{\alpha\}$. Since *NextFirst* execution interrupts as soon as something goes wrong, the network's success probability is equivalent to that of t_{leaf}: (4) reduces to (5), where $P_j(t)$ is the random function that returns the *unconditional* probability that $t_j \in T_C$ gets dynamically assigned time unit t, when following *NextFirst* protocol. $P_j(t)$ is recursively computed from t_j to t_0. In general, the *NextFirst* DP-robustness of any time event t_j is:

$$r^{\text{nf}}(j, N) = \sum_{t \in H} P_j(t). \quad (6)$$

The remaining of this section describes the computation of $P_j(t)$. We necessarily have $P_{t_0}(0) = 1$ and $P_{t_0}(t) = 0$ for $t > 0$. For any time point t_j, other than initial t_0, let

$$f_j(t) \equiv \mathbb{P}\Big\{ t = \max_{i:1..n}(t_i+l_i) \wedge t \le \min_{i:1..n}(t_i+u_i) \Big\} \quad (7)$$

be the probability that t_j may be assigned value t, when *not* considering its lower bounding constraint l_{0j}, if any. Time bounds $[l_{ij}, u_{ij}]$, $i : 1..n$ are noted $[l_i, u_i]$ for short. *NextFirst* protocol tells us t must be equal to $\max(t_1+l_1, \dots, t_n+l_n)$, except if t comes too late, as suggested by the second condition of (7). Then, now considering constraints l_{0j} as well:

$$P_j(t) = \begin{cases} \sum_{t'=0}^{l_{0j}} f_j(t') & \text{if } t = l_{0j} \\ f_j(t) & \text{if } l_{0j} < t \le u_{0j}, t \ge 0 \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The summation in the first case accounts for the situations in which $t_i+l_{ij} < l_{0j}$. Finally, the computation of $f_j(t)$ depends on the type of time point t_j which belongs to: either *transition* or *synchronisation* point.

Transition point. The time point has at most one incoming edge (i, j), either contingent or controllable, in addition to the incoming controllable edge $(0, j)$. This is true for all time points in Fig. 1 excepted t_0 and t_{11}. Under *NextFirst*, where for any controllable event, $p^d_{ij} = 1$ if $d = l_{ij}$, 0 otherwise:

$$f_j(t) = \sum_{0 \le d \le t} p^d_{ij} \cdot P_i(t-d) \ \text{ if } t \le u_{0j}, \ 0 \text{ otherwise,} \quad (9)$$

Synchronisation point. A t_j having three or more *controllable* incoming edges (e.g. t_{11} in Fig. 1) is called a *synchronisation point.* From a probability point of view:

$$f_j(t) = \mathbb{P}\Big\{ \underbrace{\bigwedge_{i:1..n} t_i+l_i \le t \le t_i+u_i}_{\alpha} \wedge \underbrace{\neg\big(\bigwedge_{i:1..n} t_i+l_i < t\big)}_{\beta} \Big\} \quad (10)$$

$$= \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i+l_i \le t \le t_i+u_i) \Big\} + \mathbb{P}\Big\{ \neg\big(\bigwedge_{i:1..n} t_i+l_i < t\big) \Big\}$$

$$- \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i+l_i \le t \le t_i+u_i \vee \neg\big(\bigwedge_{i:1..n} t_i+l_i < t\big) \Big\} \quad (11)$$

using the relation $\mathbb{P}\{\alpha \wedge \beta\} = \mathbb{P}\{\alpha\} + \mathbb{P}\{\beta\} - \mathbb{P}\{\alpha \vee \beta\}$. Let us assume that all t_i's involved in the synchronisation are ***mutually independent***, not to be confounded with that assumed on the contingent constraints T_C. We later discuss how to deal with dependency. In such case, the $\mathbb{P}\{\alpha\}$ and $\mathbb{P}\{\beta\}$ terms of (11) become straightforward to compute:

$$\mathbb{P}\{\alpha\} = \mathbb{P}\Big\{ \bigwedge_{i:1..n} t-u_i \le t_i \le t-l_i \Big\} = \prod_{i:1..n} F_i(t-l_i) - F_i(t-u_i-1) \quad (12)$$

$$\mathbb{P}\{\beta\} = 1 - \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i < t-l_i \Big\} = 1 - \prod_{i:1..n} F_i(t-l_i-1) \quad (13)$$

where $F_i(t)$ is the *cdf* of t_i: $F_i(t) = \sum_{t':1..t} P_i(t')$. If there is no upper bound constraint u_i (i.e., $u_i = \infty, i : 1..n$), then $\mathbb{P}\{\alpha \vee \beta\} = 1$. Otherwise, we have $\mathbb{P}\{\alpha \vee \beta\} =$

$$\begin{aligned} &\mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i{+}l_i \le t \le t_i{+}u_i \vee \neg\big(\bigwedge_{i:1..n} t_i{+}l_i < t\big)\Big\} \\ &= 1 - \mathbb{P}\Big\{ \bigvee_{i:1..n} t_i > t{-}l_i \vee t_i < t{-}u_i \wedge \bigwedge_{i:1..n} t_i < t{-}l_i \Big\} \\ &= 1 - \mathbb{P}\Big\{ \bigvee_{i:1..n} t_i < t{-}u_i \wedge \bigwedge_{i:1..n} t_i < t{-}l_i \Big\} \end{aligned}$$

Removing redundant conjunctions due to $t{-}u_i \le t{-}l_i$ leads to the noticeable square shaped clauses:

$$\begin{aligned} 1{-}\mathbb{P}\Big\{ &(\boldsymbol{t_1 < t{-}u_1} \wedge\ t_2 < t{-}l_2\ \wedge \ldots \wedge\ t_n < t{-}l_n\ ) \\ &\vee (\ t_1 < t{-}l_1\ \ \wedge \boldsymbol{t_2 < t{-}u_2} \wedge \ldots \wedge\ t_n < t{-}l_n\ ) \\ &\ldots \\ &\vee (\ t_1 < t{-}l_1\ \ \wedge\ t_2 < t{-}l_2\ \ \wedge \ldots \wedge \boldsymbol{t_n < t{-}u_n}) \Big\} \end{aligned} \tag{14}$$

Let A_i be the random event in which the conjunction at line i of (14) is true. The intersection of any two or more A_i's leads a conjunction of same size n. For example, $\mathbb{P}\{A_1 \cap A_2\} =$

$$\begin{aligned} &\mathbb{P}\Big\{\ (\boldsymbol{t_1 < t{-}u_1} \wedge\ t_2 < t{-}l_2\ \ \wedge \ldots \wedge\ t_n < t{-}l_n\ \ ) \\ &\quad \wedge\ (\ t_1 < t{-}l_1\ \ \wedge \boldsymbol{t_2 < t{-}u_2} \wedge \ldots \wedge\ t_n < t{-}l_n\ \ )\Big\} \\ &= \mathbb{P}\Big\{\ (\ \boldsymbol{t_1 < t{-}u_1} \wedge \boldsymbol{t_2 < t{-}u_2} \wedge \ldots \wedge\ t_n < t{-}l_n\ \ )\Big\}. \end{aligned}$$

Similarly, $A_1 \cap A_2 \cap A_i$ also leads to a A-shaped conjunction of exactly n inequalities, and so on. Since the t_i random variables are assumed mutually independent, the probability of an event $A_{I \subseteq \{1..n\}} = \bigcap_{i \in I} A_i$ is simply the product of all the probabilities of its terms. For example, for $A_I = A_1 \cap A_2$:

$$\mathbb{P}\{A_1 \cap A_2\} = F_1(t{-}u_1{-}1) \cdot F_2(t{-}u_2{-}1) \cdot \ldots \cdot F_n(t{-}l_n{-}1).$$

Using the *inclusion-exclusion principle*, we finally rewrite:

$$\begin{aligned} \mathbb{P}\{\alpha \wedge \beta\} &= 1 - \mathbb{P}\Big\{ \bigcup_{i:1..n} A_i \Big\} \\ &= 1 - \sum_{k:1..n} \left((-1)^{k-1} \sum_{\substack{I \subseteq \{1..n\} \\ |I|=k}} \mathbb{P}\Big\{ \bigcap_{i \in I} A_i \Big\} \right). \end{aligned} \tag{15}$$

Since we assumed t_i variables to be mutually independent, $\mathbb{P}\{\bigcap_{i \in I} A_i\}$ is computable as a product of $F_i(\cdot)$'s. However, what if (a subset of) the t_i's are not independent?

Proposition. Random variables $t_1, \ldots, t_n$ are dependent if they share at least one common unpredictable ancestor. A time point is a predictable ancestor of t_i *iff* its value is deterministic, and can be reached from t_i by reversing edges.

Proof: Independence hypothesis between contingent constraints implies t_i's to be mutually independent if they share no common ancestor. Now suppose: ***a)*** All common ancestors are predictable. An equivalent network is obtained by removing those and adding a constraint $[l_{0j}, \infty[$ to all remaining events, where l_{0j} is the predicted value of the closest ancestor. ***b)*** At least one common ancestor t_a is not predictable. Knowing t_i's value limits the possible realizations for t_a, which in turn influences any $t_{i'}$ having t_a as ancestor.

Corollary. In the case some contingent constraint has bounded probability distribution, *(b)* does not hold in general. Yet, we can still infer that if they do not share any common unpredictable ancestor, the events are consequently mutually independent. This is the case for t_6 and t_{10} in Fig. 1.

Imposing independence. Whenever a subset of the t_i's are potentially dependent, we impose "local independence" on them, by fixing the time value of their closest common ancestor t_a, using the law of total probability:

$$f_j(t) = \sum_{t' \in H} f_j(t \mid t_a = t') \cdot P_a(t'). \tag{16}$$

In fact, $\{t_a = 0, t_a = 1, \ldots, t_a = h\}$ is a partition of Ω. Probability $f_j(t \mid t_a = t')$ is computed after reprocessing part of the network, up to t_j, with t_a fixed to value t'. That part corresponds to all the uncommon ancestors, that is, every time point being an ancestor of at least one, but all, of the t_i's.

Computational complexity. At synchronisation time points, the complexity of computing (15), given t, depends on the maximum number L of dependent events at a synchronisation point. The *cdfs* $F(\cdot)$ being incrementally maintained, (15) requires $\mathcal{O}(L \cdot 2^L)$ operations, to be repeated at most h times through (16). Putting pieces together, the overall worst-case complexity of computing r^{nf} for a PSTN involving $m = |T|$ time events is bounded by $\mathcal{O}(mh^2 L \cdot 2^L)$.

6 Experimental Validation

We validate our approach on the same dataset as [Akmal *et al.*, 2019] , involving 452 dynamically controllable and 110 uncontrollable instances, with uniform contingent durations. Durations are continuous, whereas our method requires a discrete horizon $H = 1..h$, so we must round each time constraint $[l, u]$, in a pessimistic way: $\lceil l \rceil, \lfloor u \rfloor$ in the controllable case, $\lceil l \rceil, \lceil u \rceil$ for contingent edges. For better accuracy, we include the first D decimals of time bounds l, u at rounding. More decimals leads to better accuracy, but increases the computation times as each decimal multiplies h by 10.

Computational method. For each of the 110 uncontrollable PSTNs, we compare the r^{nf} computed values with Monte Carlo (MC) simulations, which simply record the average success rate of *NextFirst* protocol on M randomly sampled scenarios. The following table gives an overview of the computation times (min, max, mean, geometric mean) as well as the average difference $|r^{\text{nf}} - \text{MC}|$, and maximal difference dmax, between predictions and simulations:

	min	max	mean	geo	$\lvert r^{\text{nf}}$-MC$\rvert$	dmax
$M = 10^4$	0.01	0.62	0.12	0.08	0.003	0.01
$M = 10^5$	0.14	5.58	1.06	0.73	0.0007	0.004
r^{nf} D=2	0.0002	3.95	0.34	0.09	*n.a.*	*n.a.*
r^{nf} D=3	0.0002	52.9	3.85	0.18	*n.a.*	*n.a.*

We now replace the initial uniform distributions involved in the dataset by ordinary ones, randomly generated within the initial bounds. Examples are provided in Fig. 3. The following table shows the average results obtained, where we notice an increased computation time for MC, due to more expensive sampling operations on non-parametric distributions:

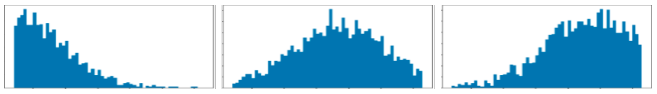

Figure 3: Examples of randomly generated ordinary distributions.

	min	max	mean	geo	$\lvert r^{\text{nf}}$-MC$\rvert$	dmax
$M = 10^4$	0.02	2.47	0.39	0.24	0.002	0.01
$M = 10^5$	0.02	20.6	3.5	2.18	0.0007	0.004
r^{nf} D=2	0.0004	3.23	0.29	0.08	*n.a.*	*n.a.*
r^{nf} D=3	0.0006	53.8	4.38	0.21	*n.a.*	*n.a.*

Experiments thus validate our equations, but also point potential limitations of our framework, in terms of computational time. When rounding time values to D=3 decimals, the resulting horizon is of h= 49300 time units in average. For some instances, rounding to 4 decimals results in unreasonable computation times (and memory usage).

***NextFirst* dispatching protocol.** We now compare our results (rounding to D=3 decimals) with that of [Akmal *et al.*, 2019], on the 110 uncontrollable instances. In the table below, line a (*resp.* b) corresponds to the number of instances for which the robustness measure (*resp.* simulations) used in [Akmal *et al.*, 2019] is of at least the probability p:

p	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
a	97	92	86	80	73	67	61	49	32	0
b	98	91	85	79	70	64	56	45	29	5
r^{nf}	**99**	**94**	**89**	**82**	**76**	**68**	59	48	**32**	**5**

The average robustness difference $a - r^{\text{nf}}$ is of -0.011, a percent of probability gain. Tested on the 452 dynamically controllable instances, *NextFirst* obtains 1.0 robustness on all of them. It means the dynamically controllable instances should not be considered as "hardly controllable", since solved by a simple dispatching protocol. Our results show that despite its simplicity, *NextFirst* achieves noticeably good robustness compared with a state-of-the-art method. Consequently, the DP-robustness of *NextFirst* provides a pertinent, close lower bound to the true DDC of a PSTN.

7 Application: Mars 2020 Rover's Task Network Structural Analysis

Our computational method allows to compute a valid lower bound on the probability of success of each time event of a PSTN, under dynamic control, following eq. (6). We exploit that ability and propose a generic *structural brittleness analysis* method (inspired by the analysis in [Vaquero *et al.*, 2019]) which computes the impact, on each and every time events of the network, and on the network itself, of increasing (or decreasing) the uncertainty of an activity a by $\alpha\%$.

As a proof-of-concept, let us analyse a typical *Mars 2020* rover (M2020) sol type, represented as a PSTN, of 18 contingent activities (36 time events). As our M2020 PSTN only (currently) involves Normal distributions, increasing (or decreasing) an activity's uncertainty amounts at modifying its standard deviation by $\alpha\%$. The following table shows how the network gets structurally affected by $\alpha = +50\%$.

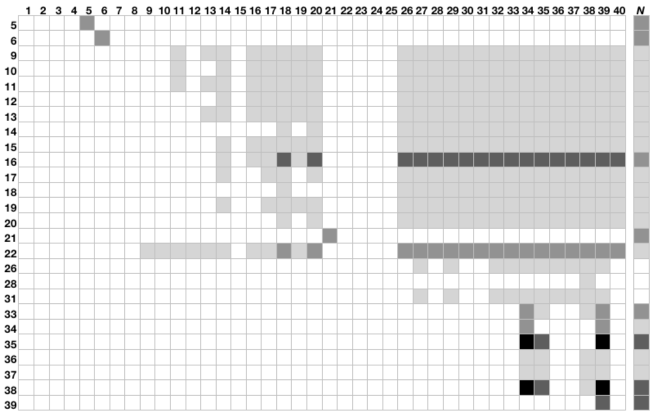

Figure 4: Structural dependency matrix of a M2020 task network of typical size: 40 activities. Empty rows are not shown. The darker the color of a cell, the bigger the robustness impact.

Cell at row i, column j, gives a amount of probability loss of activity j success when i's uncertainty is increased by 50%:

	3	4	...	12	13	14	...	18	r^{nf}
	95.4	*51.0*	...	*90.5*	*90.5*	*90.4*	...	*100*	*43.7*
3	-8.5								-3.9
4		-0.2							-0.2
...			...						
12				-9.6	-9.6	-9.6			-4.7
13					-0.0				-0.0
14						-0.3			-0.1
...							...		
18								-0.0	-0.0

Empty rows and column are not displayed. The second line of the table gives the initial success probability (%) of each activity; r^{nf} (last column) stands for the entire network's DP-robustness. In fact, the resulting matrix appears quite sparse, meaning that most activities (1-2, 5-11, 15-17) have no impact at all on the robustness of the network. Some activities (3, 4, 13, 14, 18) impact only their own robustness, although some are really unstable: 4 has only 51% success probability, but does not affect the remaining activities. Finally, activity 12 (a long duration drive) here should be considered as *structurally critical*, as it impacts others (13, 14); it also has the biggest overall impact on the network (-4.7%). This activity has a tight execution time window, which in fact, makes it brittle. This is the same kind of hardly constrained activities observed in [Vaquero *et al.*, 2019].

Finally, Fig. 4 gives an overview on the structural dependency of another M2020 task network. We notice the most critical activities as being 16, 22, 35, 38. Activities 16 and 22 are remote sensing activities (*SuperCam*). 35 and 38 are *NavCam* imaging activities. Unlike 12, they enjoy large time window and thus are likely to succeed (*e.g.* 22's uncertainty does not even impact its own success probability); however, they have a critical impact on many activities later on. These are typically the kind of activities that would not be spotted by previous brittleness analysis, unlike *e.g.* 5 and 6.

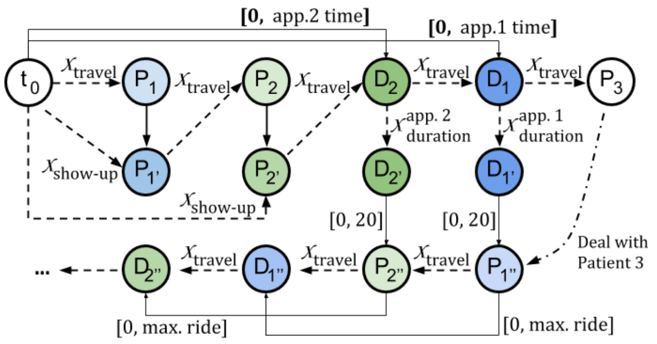

Figure 5: The vehicle first travels to Patient 1 (P1) location with a stochastic travel duration. The patient show-up time is also stochastic, and may be late. P1 is then picked up ($P_{1'}$) as soon as the vehicle arrives and the patient shows up. Then P2 is picked up ($P_{2'}$). Now P2 is dropped to her appointment (D_2), in time, which finishes at a time $D_{2'}$. Meanwhile, P1 is also dropped to her appointment. After dealing with other patients (*e.g.* P3), P1 is picked up ($P_{1''}$) at most 20 minutes after the end of her appointment. P2 ($P_{2''}$) is then picked up before dropping P1 ($D_{1''}$) at home, fulfilling a maximum ride time constraint, and so it is for P2.

8 Other Potential Application Domains

PSTNs are *not* limited to the coordination of planetary rovers. When considering ordinary distributions, it could contribute at providing new, original tools for solving various challenging real life problems. In the following applications, both PSTN and *NextFirst* seem particularly well suited.

On-demand public transportation. Consider the problem of transporting patients from their home to medical appointments [Paquay *et al.*, 2020]. The so-called dial-a-ride problem (DARP) admits a significant part of temporal uncertainty: patient delays, traffic jams, appointment durations. Each request consists in picking up a patient from home and dropping it at an appointment, and return it to home. Because the limited number of vehicles, the requests are typically mixed. The problem consists in scheduling the pickup and deliveries so that the probability of meeting all the constraints is maximized. In such context, only sampling approaches have been proposed in order to approximate the probability of success of a schedule. The PSTN formalism is particularly well suited for describing such a schedule, as illustrated in Fig. 5. Not only the PSTN formalism applies to such operational context, but also the method we propose efficiently computes the exact success probability of a given schedule.

Operations management in hospitals. Consider the problem of managing the use of operating rooms in a hospital. Whereas a number of non-urgent surgeries are known in advance, additional emergencies requiring an immediate surgery arise daily in a dynamic fashion. There is uncertainty associated with a surgery duration. Provided a limited number of rooms, hospitals must schedule their interventions in a way that any upcoming emergency can be scheduled as soon as possible. Given an a priori surgery schedule, what is the expected delay of each scheduled surgery due to the arising of emergencies? A possible PSTN representation of an operating room schedule, while considering online emergencies, is shown in Fig. 6. Given the probability of an emergency arising during a time interval, and the probability distribution of such an event duration, one can infer a contingent duration describing the additional time inquired at facing the emergency. Due to the inherent non-symmetric multi-modal nature of such distributions describing events that may or not arise, our method offers the only known efficient computation of the expected impact of online events on predefined schedules. Here, the goal is not to compute a success probability. Instead, our method can compute the exact probability distribution of each scheduled surgery's time assignment, and therefore its expected delay w.r.t. the initial schedule.

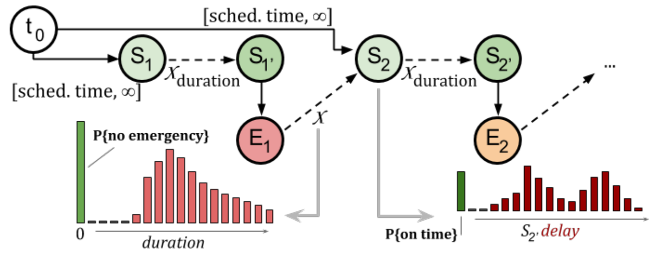

Figure 6: Operations start with scheduled surgery S_1, having uncertain duration. There is a probability that an emergency appears in the meanwhile. If it does, depending on its scheduled time, S_2 may be delayed. Our framework has the particularity to infer the time value assignment probabilities at any time event, such as S_2. This allows to compute expected delays, the probability to be on time, *etc.*

9 Conclusions

We introduced an exact approach to robustness computation in the context of dynamic control of uncontrollable probabilistic temporal networks, under discrete time assumption. By relying on simplifying operational assumptions, the computed robustness constitutes a strict lower bound on the degree of dynamic controllability under perfect dynamic assignment. Our method positions as an exact alternative to Monte Carlo simulations, by also allowing to compute the success probability of each activity of the network separately, leading to new potential applications and analysis frameworks. An open source C++ implementation of our method is available online: https://bitbucket.org/mstguillain/dprobustness.

The developed method will be useful for Mars 2020 or rovers to come. Further applications are under considerations, including actual duration data histograms (*i.e.* non-parametric, ordinary distributions) from Mars Science Laboratory, for which the results should differ from those using approximated normal distributions. We also described potential contributions to significantly different application domains, for which the only possible approach so far relied on sampling approximations (*e.g.* Monte Carlo), as our method allows to compute exact probability distributions of any time point assignment, and deals with ordinary distributions.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[Agrawal *et al.*, 2019] Jagriti Agrawal, Wayne Chi, Steve Chien, Gregg Rabideau, Stephen Khun, and Daniel Gaines. Enabling Limited Resource-Bounded Disjunction in Scheduling. *Proceedings of the 11th International Workshop on Planning and Scheduling for Space (IWPSS)*, pages 7–15, 2019.

[Akmal *et al.*, 2019] Shyan Akmal, Savana Ammons, Hemeng Li, and James C Boerkoel Jr. Quantifying Degrees of Controllability in Temporal Networks with Uncertainty. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[Bhargava and Williams, 2019] Nikhil Bhargava and Brian C. Williams. Complexity bounds for the controllability of temporal networks with conditions, disjunctions, and uncertainty. *Artificial Intelligence*, 271:1–17, 2019.

[Birge and Louveaux, 2011] John R Birge and François Louveaux. *Introduction to stochastic programming*. Springer, New York, NY, 2011.

[Brooks *et al.*, 2015] Jeb Brooks, Emilia Reed, Alexander Gruver, and James C Boerkoel Jr. Robustness in Probabilistic Temporal Planning. In *29th AAAI Conference on Artificial Intelligence*, pages 3239–3246, 2015.

[Cesta *et al.*, 1998] Amedeo Cesta, Angelo Oddi, and Stephen F. Smith. Profile based algorithms to solve multiple capacitated metric scheduling problems. In *International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 214–223, 1998.

[Chi *et al.*, 2018] Wayne Chi, Steve Chien, Jagriti Agrawal, Gregg Rabideau, Edward Benowitz, Daniel Gaines, Elyse Fosse, Stephen Kuhn, and James Biehl. Embedding a Scheduler in Execution for a Planetary Rover. In *28th International Conference on Automated Planning and Scheduling (ICAPS)*, 2018.

[Chi *et al.*, 2019] Wayne Chi, Jagriti Agrawal, Steve Chien, Elyse Fosse, and Usha Guduri. Optimizing Parameters for Uncertain Execution and Rescheduling Robustness. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[Cui and Haslum, 2019] Jing Cui and Patrik Haslum. Dynamic controllability of controllable conditional temporal problems with uncertainty. *Journal of Artificial Intelligence Research*, 64:445–495, 2019.

[Cui *et al.*, 2015] Jing Cui, P Yu, Cheng Fang, Patrik Haslum, and Brian C Williams. Optimising Bounds in Simple Temporal Networks with Uncertainty under Dynamic Controllability Constraints. In *19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 52–60, 2015.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991.

[Gaines *et al.*, 2016] Daniel Gaines, Robert Anderson, Gary Doran, William Huffman, Heather Justice, Ryan Mackey, Gregg Rabideau, Ashwin Vasavada, Vandana Verma, Tara Estlin, Lorraine Fesq, Michel Ingham, Mark Maimone, and Issa Nesnas. Productivity Challenges for Mars Rover Operations. *Proceedings of the International Conference on Automated Planning and Scheduling, Planning and Robotics Workshop (PlanRob)*, 2016.

[Kumar *et al.*, 2018] Satish T. K. Kumar, Zhi Wang, Anoop Kumar, Craig Milo Rogers, and Craig A. Knoblock. Load scheduling of simple temporal networks under dynamic resource pricing. In *32nd AAAI Conference on Artificial Intelligence*, 2018.

[Morris *et al.*, 2001] Paul Morris, Nicola Muscettola, and Thierry Vidal. Dynamic Control Of Plans With Temporal Uncertainty. In *17th International Joint Conference on Artificial Intelligence (IJCAI)*, page 494–499. Morgan Kaufmann Publishers Inc., 2001.

[Morris, 2014] Paul Morris. Dynamic controllability and dispatchability relationships. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 464–479. Springer, 2014.

[Paquay *et al.*, 2020] Célia Paquay, Yves Crama, and Thierry Pironet. Recovery management for a dial-a-ride system with real-time disruptions. *European Journal of Operational Research*, 280:953–969, 2020.

[Rabideau and Benowitz, 2017] Gregg Rabideau and Ed Benowitz. Prototyping an Onboard Scheduler for the Mars 2020 Rover. *Proceedings of the International Workshop on Planning and Scheduling for Space, IWPSS*, 2017.

[Saint-Guillain, 2019] Michael Saint-Guillain. Robust Operations Management on Mars. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*, Berkeley, CA, USA, 2019.

[Tsamardinos, 2002] Ioannis Tsamardinos. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Hellenic Conference on Artificial Intelligence*, pages 97–108, 2002.

[Vaquero *et al.*, 2019] Tiago Vaquero, Steve Chien, Jagriti Agrawal, Wayne Chi, and Terrance Huntsberger. Temporal Brittleness Analysis of Task Networks for Planetary Rovers. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[Vidal and Ghallab, 1996] Thierry Vidal and Malik Ghallab. Dealing with Uncertain Durations In Temporal Constraint Networks dedicated to Planning. In *12th European Conference on Artificial Intelligence (ECAI)*, pages 48–54. PITMAN, 1996.

[Williams *et al.*, 2009] Brian Williams, Patrick Conrad, and Julie Shah. Flexible Execution of Plans with Choice. In *19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 74–81, 2009.

[Wilson *et al.*, 2014] Michel Wilson, Tomas Klos, Cees Witteveen, and Bob Huisman. Flexibility and decoupling in Simple Temporal Networks. *Artificial Intelligence*, 214:26–44, 2014.