Journal of Artificial Intelligence Research 1 (2021) Submitted ; published

Probabilistic Temporal Networks with Ordinary Distributions: Theory, Robustness and Expected Utility

Michael Saint-Guillain MICHAEL.SAINT@UCLOUVAIN.BE
Université catholique de Louvain,
Place de l'Universite 1, 1348 Ottignies-Louvain-la-Neuve, Belgium

Tiago Stegun Vaquero TIAGO.STEGUN.VAQUERO@JPL.NASA.GOV
Steve A. Chien STEVE.A.CHIEN@JPL.NASA.GOV
Jagriti Agrawal
Jet Propulsion Laboratory, California Institute Technology,
4800 Oak Grove Dr, Pasadena, CA 91109 USA

Jordan Abrahams JABRAHAMS@HMC.EDU
Harvey Mudd College,
301 Platt Boulevard, Claremont, CA 91711

Abstract

Most existing works in Probabilistic Simple Temporal Networks (PSTNs) base their frameworks on well-defined, parametric probability distributions. Under the operational contexts of both strong and dynamic control, this paper addresses *robustness measure* of PSTNs, i.e. the execution success probability, where the probability distributions of the contingent durations are *ordinary*, not necessarily parametric, nor symmetric (*e.g.* histograms, PERT), as long as these can be discretized. In practice, one would obtain ordinary distributions by considering empirical observations (compiled as histograms), or even hand-drawn by field experts. In this new realm of PSTNs, we study and formally define concepts such as degree of weak/strong/dynamic controllability, robustness under a predefined dispatching protocol, and introduce the concept of PSTN *expected execution utility*. We also discuss the limitation of existing controllability levels, and propose new levels within dynamic controllability, to better characterize dynamic controllable PSTNs based on based practical complexity considerations. We propose a novel fixed-parameter pseudo-polynomial time computation method to obtain both the success probability and expected utility measures. We apply our computation method to various PSTN datasets, including realistic planetary exploration scenarios in the context of the *Mars 2020* rover. Moreover, we propose additional original applications of the method.

1. Introduction

Temporal networks formalize the arrangement and inter-dependencies of tasks, or activities, that compose an operational plan. In a simple temporal network (STN), activities are modeled as a finite set of time events, such as start and end times. In practice, some activity durations, considered as *contingent*, remain unknown beforehand and are revealed during execution (decided by nature). When some stochastic knowledge on the uncertain durations exists, one can model it as (estimated) probability distributions, leading to the extending concept of probabilistic STN, or *PSTN*. Solving a PSTN then amounts at finding an assignment of time values to events, such that assigned values together with observed ones fulfil all the constraints between events (*e.g.*, end of task A must happen between 10

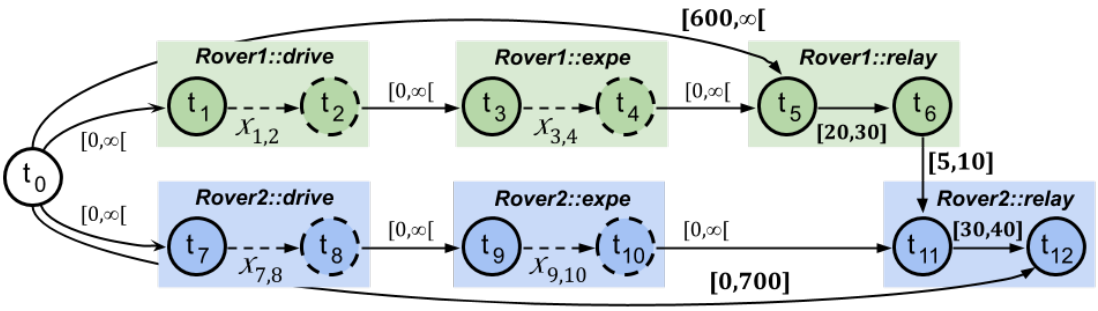

Figure 1: A simplified hypothetical sol on Mars for two planetary rovers, encoded as a PSTN. Bold: controllable. Dashed: contingent.

and 20 minutes before the beginning of B). Whenever such assignment exists, a network is said to be *controllable*. When the operational assumptions enable it, the assignment 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 as violating some of the temporal constraints during execution. When there is uncertainty with respect to temporal constraint violation, it is critical to evaluate how likely a PSTN will lead to a successful execution to help mitigate unsafe operations. This evaluation is called the Degree of Dynamic Controllability (DDC) (Akmal, Ammons, Li, & Jr, 2019). The DDC should not be confused with the *robustness*, which characterizes the likelihood of success of a PSTN under a specific reoptimization scheme, also known as *dispatching protocol*, or execution strategy. The DDC then represents robustness under a perfect (*i.e.* optimal, in the multistage stochastic sense) online reoptimization scheme. Whereas computing the exact DDC is intractable in practice (NP-hard), it may still be possible to provide some guarantees, in terms of (lower) bounds or approximations.

The current literature proposes DDC computation methods for PSTNs that involve *unimodal distributions only* (*e.g.* uniform or normal). In realistic cases however, this might be an unrealistic assumption/approximation, especially for activities which duration distributions are asymmetric by nature (*e.g.* vehicle driving operations). In certain domains (*e.g.* Mars rover operations), deciding for a particular parametric distribution could be problematic. In that context, exploiting the raw observations directly, namely the histograms, could avoid making wrong assumptions. Existing DDC computation methods are not applicable in these non-symmetric, non-parametric distributions, except for Monte Carlo simulation, which admits ordinary distributions at the price of coming without any guarantee.

Contributions.

In this study, we proposed the first efficient DDC computation method capable of dealing with PSTNs with *any possible ordinary probability distributions*. Our method computes an *exact lower bound* on the DDC of a PSTN, hence being the first to provide a guarantee to never overestimate this probability of success, by considering the robustness under a specific *dispatching protocol* – a dynamic decision scheme simpler than performing perfect reoptimization, a concept we formally define. 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 (Vaquero, Chien, Agrawal, Chi, & Huntsberger, 2019) and consequently supports operators or users to identify activities that highly impact the success probability of other activities, and the robustness of the entire network. Remark however that, whereas the majority of PSTN work considers continuous probability distributions, our method requires these to be discretized, or their cumulative distribution functions (CDFs) to be evaluated at discrete values.

Our robustness computation assumes that no activity may be interrupted. However, in certain applications, the activities can be interrupted before completion following predefined cutoff times, without entailing the entire network (nor the subsequent activities) execution failure. In such a context, some activity may be more expensive to interrupt than others. By further assigning a utility value to each activity (the higher the utility, the more expensive), we introduce the concept of *expected utility* of a PSTN, which measure how likely important activities are to be eventually interrupted (*i.e.* reaching their cutoff times). If all activities has unit utility, then the expected utility is simply the expected number of uninterrupted tasks by the end of the online network execution. Herein, the expected utility of a network is a complementary metric to its DDC and robustness. We show how this expectation can also be computed using our method.

We extend the concepts of DDC and DSC (degree of strong controllability) to PSTNs, and introduce the first formal definitions of both, in the case of discrete time horizon. We also introduce and define the degree of weak controllability (DWC) measure. Finally, from these definitions we deduce remarkable fundamental inequalities. We also propose new theoretical tools, namely dynamic controllability sub-levels, for better characterizing dynamic controllable PSTNs. Finally, we provide a unified view of the research landscape on DDC and DSC measurement in probabilistic simple temporal networks.

The results obtained on a benchmark from the current state-of-the-art literature validate the soundness of our method, on both parametric and ordinary distributions. As part of the method validation in PSTNs with parametric distributions, we compare our approach against state-of-the-art DDC approximation methods which aim at getting as close as possible to the true DDC of a PSTN, at the risk of over-estimating it. The lower bounds obtained from our method tend to be higher than the approximations computed by the state-of-the-art methods, thus being closer to the true DDC value, with the guarantee of not exceeding it. In those cases, the state-of-the-art methods are therefore too conservative. Whereas in the cases where our computed value is below the state-of-the-art approximation, the existing methods may be overestimating the DDC.

We apply our method to the real case study of the Mars 2020 rover's task networks, where activities duration are described as ordinary distributions inferred from historical observations gathered during previous Mars rover operations. In this application, we discuss the use of limited historical observations (in the form of sparse histograms) against the use of estimated parametric distributions to represent uncertainty on the activity durations. Moreover, we propose a new method for identifying structural bottlenecks in the temporal brittleness analysis of Mars 2020 rover's task networks.

This study extends that of (Saint-Guillain, Stegun Vaquero, Agrawal, & Chien, 2020) conference paper as follows: i) we provide a thorough description of the theoretical contribution on the DDC lower bound computation; ii) we generalize our theory to propose a

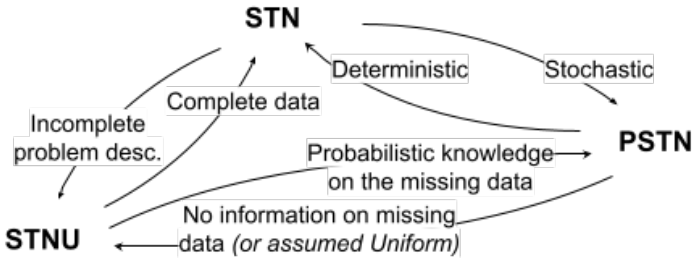

Figure 2: Simple temporal networks variants with respect to uncertainty.

unified view not only on STNU/PSTN dynamic controllability, but also strong and weak controllability, and prove various relations and bounds between these; iii) we expand the theoretical contributions to include new dynamic controllability sub-levels; iv) we introduce new theory to compute the expected utility of a PSTN; and v) expand the experimental validations and applications accordingly.

2. Temporal Networks, Policies and Dispatching Protocols

2.1 STN.

Simple Temporal Network is a popular formalism for temporal constraint reasoning (Dechter, Meiri, & Pearl, 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 $t_i \in T$. A schedule is consistent if it satisfies all the constraints of the network. A STN is *consistent* if it admits a consistent schedule. In practice, a time point in a STN often stands for either the start or the end of a particular *activity*. Considering the end point t_j of some activity, constraint $c(t_i, t_j)$ then represents the valid bounds for the activity duration defined by time points (t_i, t_j), whereas $t_j - t_i$ gives the duration set up by a specific schedule. Alternatively, when t_j stands for an activity start point, $c(t_i, t_j)$ constrains how long the activity beginning can be delayed from previous (usually end of activity) time point t_i, and $t_j - t_i$ gives its actual value.

Definition 1 (STN schedule)**.** *In the STN context, a schedule $x_1, \ldots, x_n$ is an assignment (mapping) $T \to \mathbb{R}^n$ of all time points $t_i \in T$ of the network.*

2.2 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 temporal constraints, such as activity durations (Tsamardinos, 2002; Fang, Yu, & Williams, 2014; Brooks, Reed, Gruver, & Jr, 2015; Santana, Vaquero, Toledo, Wang, Fang, & Williams, 2016). 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 by 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. Whereas PSTN assume a stochastic knowledge on each contingent activity duration, the case where one has no information at all on the unpredictable durations is called a STN with Uncertainty: **STNU** (Vidal & Ghallab, 1996). Provided only lower and upper bounds on a contingent duration $(t_j - t_i)$, a common usage is to consider it uniform: $X_{ij} \sim U(l_{ij}, u_{ij})$. Despite their initial definition, the term STNU evolved to also include the subset of PSTNs that involve uniform distributions only, although technically speaking STNU is not a particular case of PSTN in general. This is illustrated in Fig. 2.

Definition 2 (PSTN schedule)**.** *In the PSTN context, a schedule $x_1, \ldots, x_n$ is an assignment (mapping) $T_E \to \mathbb{R}^n$ of all executable time points $t_i \in T_E \subseteq T$ of the network.*

Illustrative example: rover operations. An hypothetical example of Mars rovers operations, represented as a PSTN, is depicted in the Fig. 1. It shows a modified version of that provided in (Santana et al., 2016), with sol duties for two Mars planetary rovers. 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. Note that unrestricted time windows $[0, \infty[$ are usually not shown in the figures. 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, the duration of the 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 $c(t_1, t_2) \in C_C$, $t_2 \in T_C$, whereas t_{12} has a deterministic deadline represented by time window $[0, 700]$.

PDTNs versus PSTNs. Temporal networks constitute expressive tools for modelling operational plans, from human operated projects to planetary rover operations. Yet, a PSTN aims at formally stating only one of the many possible structural configurations of the network. In the PSTN of Fig. 1 for example, *Rover1* was chosen first to relay. However, another solution would have been to relay data from *Rover2* first, hence leading to a slightly different PSTN. A realistic Mars rover sol type may involve around 40 activities per rover (Chi, Agrawal, Chien, Fosse, & Guduri, 2019), which in many cases allow different ordering the activities. In this context, one may come up with as many different PSTNs as there are possible orderings of the tasks assigned to each rover, or alternatively use the Probabilistic Disjunctive Temporal Network (PDTN) formalism to consider explicit representation of choices or alternatives. PDTNs extend the formalism of PSTN by enabling, with the use of disjunctive temporal constraints, the encoding of exponentially many interpretations, in the form of PSTNs, within one single PDTN. A disjunctive temporal version of our rover example may then be obtained by removing constraint between t_6 and t_{11} and adding

the disjunctive constraint: *"either t_{11} happens after t_6, or t_5 happens after t_{12}"*. The resulting PDTN encodes our two possible PSTNs at once. The PDTN formalism is a very powerful modeling tool, even more expressive than combinatorial problems such as job-shop scheduling, as it enables to express contexts which cannot be described otherwise, such as stating *"if C has not already been completed at the time A is started, then the starting of A must happen after the completion of B"*. A recent study of PDTNs in the context dynamic control can be found in (Cimatti, Micheli, & Roveri, 2016). In this work, however, we focus on PSTN only, leaving robustness analysis for PDTN for future work.

Relation to Job-Shop Scheduling. We already remarked that PDTNs generalize job-shop scheduling problems (JSP). Just as a PDTN encodes a set of (exponentially many) PSTNs, a solution to a (probabilistic) JSP can be encoded as a PSTN. A solution to a JSP, and more specifically a JSP with probabilistic durations (Beck & Wilson, 2007), usually consists in an assignment and ordering of the tasks only, that is, without necessarily assigning start time values to the tasks. In fact, in such context each task is assumed to start as soon as possible, which exactly corresponds to the *NextFirst* dispatching protocol on which we focus in this paper. In other words, formulated as a PSTN, our method computes the probability of success of any solution to a probabilistic JSP.

2.3 Policies and Dispatching Protocols

Operational contexts such as space missions usually pose computational and power limitations on 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 yield and time. Such approach is currently operating *Curiosity* rover, with static schedules that overestimate processing times by 30% in average (Gaines et al., 2016b) to account to execution uncertainty. Let Ω^N be the set of all possible realizations of the random contingent edges' duration in a PSTN N. A trivial approach to avoid both static scheduling and online reoptimization is to precompute particular schedules for each possible situation that may arise, leading to a *policy*:

Definition 3 (Policy)**.** *A policy for a PSTN N is a mapping $\Omega^N \rightarrow \mathbb{R}^n$ that associates a schedule to each scenario from Ω^N.*

Naturally, the size of Ω^N is usually problematic. Instead, Perseverance (M2020) rover is 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 & Benowitz, 2017). Due to computational limitations, such online decisions must remain very light, thus following a predefined *strategy*: a *dispatching protocol* (DP). In particular, a DP usually aims at avoiding costly online reoptimizations. We derive the concept of DP from that of recourse strategy in stochastic programming (Birge & Louveaux, 2011).

Definition 4 (Dispatching protocol)**.** *Let N^t be the PSTN resulting from the execution of the network N up to current time t (with $N^0 = N$), thus including fixed values for the subset of already executed time points. Let $T_E^t \subseteq T_E$ the remaining executable time points. A dispatching protocol (DP) returns a valid assignment of a subset $\Gamma_E^t \subseteq T_E^t$ of the remaining $n = |T_E^t|$ executable time points, consistent w.r.t. previous observations and decisions:*

$$DP(N^t) : \Gamma_E^t \rightarrow \mathbb{R}^m, m \leq n$$

Note the fundamental difference between a dispatching protocol, and its particular case, a policy. When $\Gamma_E^t = T_E^t$, *i.e.* when the DP always assigns values for all remaining executable time points, then it is (a compact representation of) a policy. Otherwise, the DP postpones part of the future decisions to be made, and we have $\Gamma_E^t \subset T_E^t$. In that case, the underlying decision algorithm may be described as greedy.

We distinguish three families of DP's, depending on the complexity of DP$(\cdot)$. First, the case where DP$(\cdot)$ is constant time directly corresponds to a *static/strong* schedule $s = x_1, \ldots, x_n$. 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, with usually $\Gamma_E^t = T_E^t$.

Finally, the case of a (non-constant) polynomial-time DP$(\cdot)$ allows agents to *dynamically adapt the schedule*, while being *computationally limited*. For example, Rabideau and Benowitz (Rabideau & Benowitz, 2017) describe an average $\mathcal{O}(n^2)$ quadratic DP$(\cdot)$ protocol to be computed by the onboard scheduler in the Mars Perseverance rover, in order to adapt decisions online (*i.e.* $\Gamma_E^t = T_E^t$) based on observations and pre-optimized parameters (Chi et al., 2019). Brooks et al. (Brooks et al., 2015) consider a linear time protocol to make partial decisions ($\Gamma_E^t \subset T_E^t$), called *NextFirst* protocol, which we describe in Sec. 3.4.

3. Controllability, Robustness and Utility

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, especially in highly uncertain environments. In fact, if the PSTN of Fig. 1 involves some probability distribution with unbounded tail, then such perfect schedule does not exist. Instead, we refer to a *dispatching protocol* which maps observations to decisions during execution, hence deciding the schedule online. Under uncertain activity durations, such a greedy dynamic approach would raise the following questions: 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? What should we expect with respect to critical and non-critical activities being interrupted during execution?

3.1 Controllability levels

The traditional concept of consistency from STN is not directly applicable to PSTNs or STNUs 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 consistent schedules 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.

In this section, we introduce formal definitions for various fundamental concepts of PSTNs, and deduce new theoretical results from it. $\Phi^{\mathcal{P}}(N, \xi)$ returns 1 if following protocol $\mathcal{P} \in DP(N)$ in situation ξ leads to a successful execution, zero otherwise. In particular, $\Phi^s(N, \xi)$ indicates whether the static schedule $s = x_1, \ldots, x_n$ is consistent in scenario ξ.

Strong controllability. An PSTN is said to be *strongly controllable* (SC) (Vidal & 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:

$$N \text{ is } SC \quad \Leftrightarrow \quad \exists s \in \mathbb{R}^n : \ \forall \xi \in \Omega^N, \ \Phi^s(N, \xi) = 1$$

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.

Dynamic controllability. A more practical level would be *dynamically controllable* (DC) (Morris, Muscettola, & Vidal, 2001; Morris & Muscettola, 2005; Morris, 2014), in which we check whether there exists a dispatching protocol such that, at any time during execution, the partial sequence executed so far extends to a consistent schedule, whatever durations remain to be observed:

$$N \text{ is } DC \quad \Leftrightarrow \quad \exists \mathcal{P} \in DP(N) : \ \forall \xi \in \Omega^N, \ \Phi^{\mathcal{P}}(N, \xi) = 1$$

It requires the agent to be able to determine, in a dynamic fashion, a valid assignment of all remaining executable time points, based on observed past contingent ones, without violating any future temporal constraints. Remark that having the right protocol $\mathcal{P}$ for $N = N^0$ is sufficient, as it trivially implies the existence of the protocol for $N^1, \ldots, N^h$.

Weak controllability. Finally, weak controllability stands for a somehow less practical property. A PSTN is said to be *weakly controllable* (WC) *iff* for each possible situation, there exists a valid specific schedule.

$$N \text{ is } WC \quad \Leftrightarrow \quad \forall \xi \in \Omega^N, \ \exists s \in \mathbb{R}^n : \ \Phi^s(N, \xi) = 1$$

Here, each such situation is thus considered to be perfectly known in advance, whereas DC accounts for the fact that information is actually revealed in a dynamic fashion. From a stochastic programming point of view, the WC formulation then amounts at solving the DC stochastic multistage problem while relaxing the underlying inherent nonanticipativity constraints (Shapiro, Dentcheva, & Ruszczyński, 2014), that is, while losing the consistency between online decisions and the moment each piece of information is revealed. This will be further analyzed in the following section. Finally, remark the resemblance between the definitions of SC and WC. From the ordering of the logical quantifiers, we directly find that SC implies WC.

3.2 Robustness and Degrees of Controllability

As pointed out in Sec. 2, in practice highly uncertain environment often makes temporal network uncontrollable. When a PSTN cannot be proven dynamic (*resp.* strong) controllable, then we might most probably be interested in determining the degree of dynamic (*resp.* strong) controllability which aims at measuring how far the network actually is from being so. Whereas controllability checking has been proven polynomial in many cases (Bhargava & 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) as well as the *degree of strong controllability* (DSC) of a STNU are introduced. When considering uniform distributions only, the DDC (*resp.* DSC) is defined as the proportion of contingent edges realizations in which the temporal network remains dynamically (*resp.* strongly) controllable.

Note that there is a possible confusion (the error was made in Saint-Guillain et al. (2020), and probably others) between the terms DDC and robustness. The *robustness*, introduced in (Brooks et al., 2015), represents the success probability under a predefined (usually suboptimal) dispatching protocol. In other words, the DDC is one particular robustness measure on a PSTN N, corresponding to the particular dispatching protocol that performs optimally under uncertainty for N. These concepts, which are at the core of the current paper, are defined hereafter.

In this section, we propose to extend the concepts of DDC and DSC to PSTNs, and introduce the first formal definitions of it, in the case of discrete time horizon. We also introduce the degree of weak controllability (DWC) measure. Finally, from these definitions we deduce remarkable fundamental inequalities.

Assumptions and notations. From now on, we assume a discrete time horizon $t = 1, \ldots, h$. Depending on the modeling choices, the decisions to be taken by the online scheduler will be either represented as **(i)** a vector $x^1, \ldots, x^n$ of $\mathbb{R}^n$, hence a schedule, or **(ii)** more generally as a vector $\mathbf{x} = x^1, \ldots, x^h$ of $\mathbb{R}^h$, in which it represents the decisions taken at each and every time unit of the horizon, from which the schedule can be trivially deduced. Henceforth, the indicator function $\Phi^{\mathbf{x}}(N, \xi)$ is assumed to return 1 iff the schedule deduced from $\mathbf{x}$ is consistent in scenario ξ. Similarly, a timewise representation of the uncertainty will sometime be adopted, with a set of possible scenarios Ω^h instead of Ω^N. A realisation of Ω^h is then a sequence $\xi = \xi^1, \ldots, \xi^h$ of outcomes. When necessary, we designate by $\xi^{t..t'}$ the sequence of outcomes of scenario ξ from time t to time t', and to decisions $x^t, \ldots, x^{t'}$ as $x^{t..t'}$. Operator $\mathbb{E}_{\xi^t}[\ \cdot\ ]$ designates the expectation over random variable ξ^t, conditionally to history $\xi^{1..t-1}$.

3.2.1 Robustness

The so-called *robustness* (Brooks et al., 2015) measures the success probability of a network N under a specific dynamic dispatching protocol $\mathcal{P}$. In order to stress the dependence with a given dispatching protocol, and therefore avoid the confusion with the DDC metric, to "robustness" we sometimes prefer the term "DP-robustness", $r^{\mathcal{P}}(N)$ for short:

$$r^{\mathcal{P}}(N) = \sum_{\xi \in \Omega^N} \mathbb{P}\{\xi\}\ \Phi^{\mathcal{P}}(N, \xi) \tag{1}$$

Given a fixed scenario ξ, a PSTN reduces to a regular STN as the contingent edges get assigned fixed durations, and $\Phi^{\mathcal{P}}(N, \xi)$, the execution of protocol $\mathcal{P}$ on scenario ξ, can be checked (*i.e.* "simulated") in linear time. Nonetheless, the computation of (1) remains intractable in practice, as the size of Ω^N grows exponentially with the number of contingent edges. This motivates all kinds of sampling based methods, such as Monte Carlo, which therefore restricts the summation in (1) to a limited subset of Ω^N.

Degree of Strong Controllability. We saw that using static schedule, thus in the context of strong control, amounts at exploiting a very simple (and strict) constant time dispatching protocol. Determining the DSC is then equivalent to the problem of finding a static schedule maximizing its DP-robustness:

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

where $r^s(N)$ designates the success probability of a static schedule $s = x_i^{1..h}$, $s \in \mathbb{R}^n$. We directly see that the maximization in (2) in fact computes the DSC, since it gives the success probability of the best possible static schedule, in other terms *"how far is the network from being SC"*.

Degree of Dynamic Controllability. A remarkable particular case of DP-robustness is that of the DDC. The DDC refers to the robustness under perfect reoptimization, that is, using an optimal dispatching protocol, able to always determine the best possible decisions based on past outcomes and remaining uncertainty. It corresponds to the true probability of succeeding, assuming unlimited computational power and time. In this case, such optimal decision system must necessarily solve the following multistage stochastic program (3)

$$\mathrm{DDC}(N) = \mathbb{E}_{\xi^1}\Big[\max_{x^1} \mathbb{E}_{\xi^2}\Big[\max_{x^2} \mathbb{E}_{\xi^3}\Big[\ldots \max_{x^{h-1}} \mathbb{E}_{\xi^h}\Big[\max_{x^h} \Phi^{x^{1..h}}(N, \xi)\Big]\ldots\Big]\Big]\Big] \tag{3}$$

According to previous decisions $x^{1..t-1}$, history $\xi^{1..t-1}$ and current outcome ξ^t, a decision at stage t is computed in order to maximize the probability that the partial schedule $x^{1..t}$ extends to a consistent full schedule. At time $t = 0$ for example (before the beginning of the operations), the DDC is then the expectation, over all the possible outcomes for ξ_1 at first time unit $t = 1$, of the expected value of the best possible response to the outcome.

The nested expectations in (3) form a tree structure, well known as the *scenario tree*, as illustrated in Fig. 3 (top). Each path of the tree constitutes a possible scenario realization, a sequence $\xi = \xi^{1..h}$. To each node is associated a decision variable x^t, representing the optimal decisions at time t depending on the current history, and maximizing the expected value $\mathbb{E}_{\xi^{t+1}}[\max_{x^{t+1}} \ldots]$ of the subsequent optimal decisions at time $t+1$, and so on until time h is reached. Whereas checking dynamic controllability has been proven polynomial (Morris & Muscettola, 2005; Nilsson, Kvarnström, & Doherty, 2014), the scenario tree illustrated in Fig. 3 (top) clearly suggests NP-hardness for determining the DDC of a network.

As for the DSC, up to now the only available definition of DDC for PSTNs is the non formal one *"how far is the network from being DC"*. Recall that being DC means "there exists an execution strategy such that, at any time during execution, the partial sequence executed so far extends to a consistent schedule". The DDC is then the robustness of the protocol being the best at ending up with a consistent schedule when it is in fact possible. If we consider the recursion in (3) from the innermost maximization term, we see that the last decision x^h finds a consistent schedule, if it exists, based on scenario $\xi = \xi^{1..h}$. The decision at time $h-1$ is is necessarily the one that maximizes the probability that, after ξ^h realizes, $\max x^h$ extends to a consistent schedule. In turn, x^{h-2} is computed in light of the current state and remaining uncertainty ξ^{h-1} and ξ^h, and so on back to initial decision x^1. The multistage formulation (3) is therefore a valid definition for the DDC.

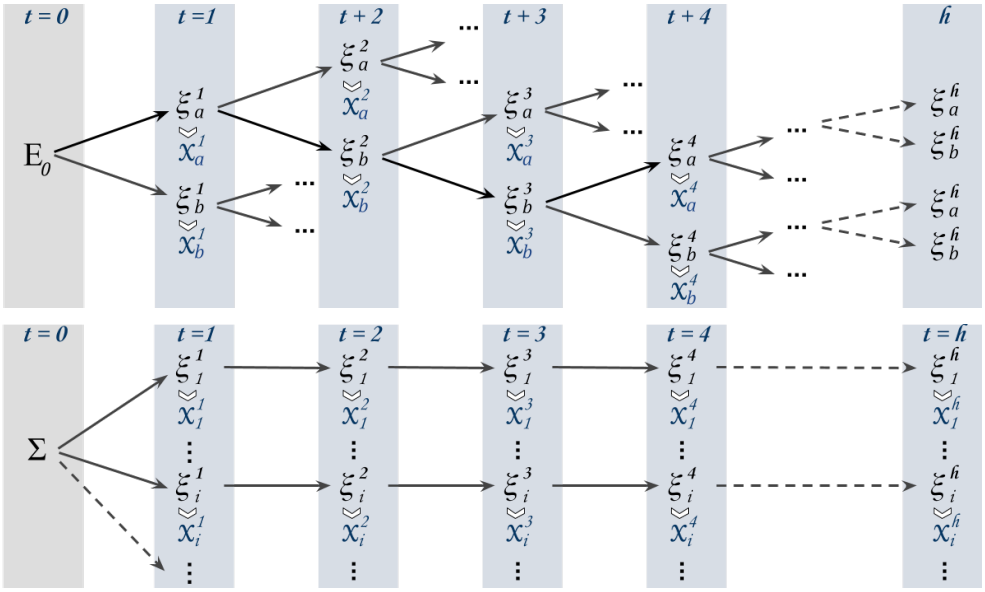

Figure 3: **Top:** Tree structure of the problem. For simplicity, the random variable has only two possible outcomes (denoted a and b) at each period, leading to 2^{h-1} leaf nodes, and as many scenarios. Implicit nonanticipativity constraints: decisions x_a^4 and x_b^4 must necessarily share the same previous decisions $x_a^1, \ldots, x_b^3$. **Bottom:** Tree structure of the problem under a scenario-wise formulation (4). Explicit nonanticipativity constraints (5) link scenarios to each others, forcing nodes that correspond to the same node in (top) to share the same decisions.

Note that the order of the expectation and maximization operators in (3), namely $\mathbb{E}_{\xi^1}, \max_{x^1}, \mathbb{E}_{\xi^2}, \ldots$, may seem a bit odd to the reader used to multistage stochastic programs. In fact, this particular order comes from the assumption that we always observe the current (*i.e.* at time t) random outcome ξ_t before making decision x_t. For example, at time $t = 0$ (before the beginning of the operations), the DDC is then the expectation, over all the possible outcomes at first time unit $t = 1$, of the expected value of the best possible response. This implies that we do not consider offline decisions. At time t however, decision x_t is made in order to maximize its (expected) value $\mathbb{E}_{\xi^{t+1}}[\ldots]$. The online decision at time t is then defined by $x = \text{argmax}_t \mathbb{E}_{\xi^{t+1}}[\ldots]$. This modeling choice is necessary to define the *DDC as a property of the network*, instead of a property of an offline decision (which, on the contrary, is by definition the case of the DSC).

Solving problem (3) is impossible in practice, as it requires solving the maximization problem at each of the exponentially many nodes of the scenario tree. This immediately suggests two possible approximation schemes, either (a) limiting the branching factor by sampling a restricted number of children generated from a node, and (b) simplify the decision choice at each node, for instance, by choosing randomly. Both (a) and (b) are compatible, and with a few additional techniques (such as back propagation) one would end up with a Monte Carlo Tree Search (MCTS) algorithm. In the case of the DDC, such approach has the

advantage of performing its simulations without being restricted by a specific dispatching protocol. In other words, approximating the expected value of each decision node would directly approximates the behavior of an optimal online scheduler, contrary to classical sampling based approaches which approximate a particular (limited) dispatching protocol.

Contrary to (3), the definition of the the robustness in (1) is a "scenario-wise" formulation, which decomposes the problem in amongst the set Ω^h of scenarios, as depicted in Fig. 3 (bottom). A formulation much closer to (1) can be obtained by reformulating the multistage stochastic program (3) as its two-stage equivalent program (Shapiro et al., 2009):

$$\mathrm{DDC}(N) = \sum_{\xi_i \in \Omega^h} \mathbb{P}\{\xi\} \max_{x_i^1,\ ...,\ x_i^h} \Phi^{x_i^1,\ ...,\ x_i^h}(N, \xi_i) \tag{4}$$

$$\text{s.t. } \forall t' : 1..h : \quad \xi_i^{1..t'} = \xi_j^{1..t'} \quad \Rightarrow \quad x_i^{1..t'} = x_j^{1..t'} \tag{5}$$

where x_i^t is the decision associated to node ξ_i^t of the scenario decomposition depicted in Fig. 3 (bottom). The maximization in (4) returns 1 if a consistent schedule exists for a scenario ξ_i, zero otherwise. Constraints (5) express the *nonanticipativity property*, by stating that if two sequences of outcomes $\xi_i^{1..t'}$ and $\xi_j^{1..t'}$, namely two prefixes of scenario, are identical (*i.e.* ξ_i and ξ_j belong to the same branch up to time t' in the scenario tree, top of Fig. 3), then the associated decisions must also be identical. In other words, a decision can only be taken in light of past realizations, not future ones! The only available information about the future are the probability distributions of remaining random variables, not their realizations. Note that in formulation (3), the nested shape of the expectations implicitly enforces these nonanticipativity constraints, which naturally prevents the scheduler from time travel to modify past decisions.

From a modeling point of view, this scenario-wise formulation clearly define the DDC as not only a property of the network, but more specifically *of the uncertainty on that network*, whereas the dispatching decisions are only consequences of particular realizations.

Degree of Weak Controllability. By relaxing the nonanticipativity constraints, that is by considering the summation (4) without (5), we obtain the total mass of scenarios for which knowing the future leads to a successful execution. Following the definition of WC in Sec. 3.5, the *degree of weak controllability* (DWC) can therefore be formulated as:

$$\mathrm{DWC}(N) = \sum_{\xi_i \in \Omega^h} \mathbb{P}\{\xi\} \max_{s}\ \Phi^s(N, \xi_i) \tag{6}$$

where $s = x_i^1,\ \ldots,\ x_i^h$, which is literally the probability mass of all the favorable scenarios in the definition of WC. Since the maximization in (6) returns 1 iff N is consistent in scenario ξ_i, the DWC is the probability that N reveals to be consistent afterwards, *i.e.* the probability of succeeding the execution when the scenario can be fully predicted in advance.

3.2.2 Basic inequalities

Given a dispatching protocol $\mathcal{P}$, and because an optimal scheduler is by definition at least as likely to succeed than an arbitrary $\mathcal{P}$, the robustness under dispatching protocol $\mathcal{P}$ necessarily constitutes *a lower bound on the true DDC* of a network. In particular and from the definition of DSC in (2), for any static schedule s we have $r^s(N) \leq DSC(N)$.

Since strong controllability implies dynamic controllability, and because (6) is a relaxation of (4)-(5), putting everything together gives the two relations:

$$\forall \mathcal{P} : \qquad r^{\mathcal{P}}(N) \leq \mathrm{DDC}(N) \leq \mathrm{DWC}(N) \tag{7}$$

$$\forall s \subseteq \mathbb{R}^n : \qquad r^{s}(N) \leq \mathrm{DSC}(N) \leq \mathrm{DDC}(N) \leq \mathrm{DWC}(N) \tag{8}$$

The left side of the relation (7) constitutes the theoretical basis for one of the main contribution of this paper, which is the computation of a lower bound on the DDC by using a specific protocol $\mathcal{P}$. The remaining relations deduced in this section, including (7) and (8) which clearly suggest a method for bounding the DDC from above, using the DWC, are only of theoretical interest for now and left for further investigation.

3.2.3 The Cost of Uncertainty

In the stochastic programming literature, part of these relations are well known to carry an important meaning. In particular, $\mathrm{DWC}(N) - \mathrm{DDC}(N)$ is known as the *expected value of perfect information* (EVPI), which represents the expected gain of being able to predict the future, the oracle's price. In (Birge & Louveaux, 2011), the authors describe the EVPI as "the maximum amount a decision maker would be ready to pay in return for complete information about the future".

Now suppose that instead of trying to solve the PSTN in light of all its uncertainty, we only consider the particular scenario $\overline{\xi}$ in which all the contingent durations realize to their the expected (mean) values. We thus end up with in a deterministic STN, for which computing a consistent schedule $s(\overline{\xi})$, if it exists, is not hard. The *expected result of using* $s(\overline{\xi})$ in a stochastic context is then

$$\mathrm{EEV}(N) = \sum_{\xi \in \Omega^N} \mathbb{P}\{\xi\} \; \Phi^{s(\overline{\xi})}(N, \xi) \; , \tag{9}$$

$$\text{with} \quad s(\overline{\xi}) = \max_{s} \left\{ \Phi^{s}(N, \overline{\xi}) \right\} \tag{10}$$

Another interesting measure is $\mathrm{DDC}(N) - \mathrm{EEV}(N)$, known as the *value of the stochastic solution* (VSS), which literally indicates the expected gain of taking uncertainty into account, instead of simply assume average durations. Eventually, the most trivial and probably important result from stochastic programming, which justifies the domain itself, is that both the EVPI and the VSS are always non-negative. This leads to a third bounding inequality in addition to (7) and (8):

$$\mathrm{EEV}(N) \leq \mathrm{DDC}(N) \leq \mathrm{DWC}(N) \tag{11}$$

The EEV therefore appears as another potentially interesting method for computing a lower bound on the DDC. On this paper however, we focus on the more promising approach of computing a lower bound from relation (7), and leave the EEV for further investigation.

3.3 DP-utility

In the previous sections, we discussed the probability of succeeding in executing all the activities of a network. Depending on the context, one may also be interested in the

expected number of activities that can be successfully achieved until one failure occurs. Similarly, whenever each task is assigned an utility or weight, one might be interested in the expected amount of utility to be reached by the end of the execution, or until a time constraint gets violated and prevents the execution of subsequent tasks. Whereas the DP-robustness of a network gives the probability of executing it successfully through the end, the expected utility further describes how far this execution is likely to be conducted.

We hence propose another relevant indicator: the *DP-utility* of a PSTN N under a predefined dispatching protocol $\mathcal{P}$. Suppose $U^{\mathcal{P}}(N,\xi)$ is the resulting total utility achieved by protocol $\mathcal{P}$ in scenario ξ. The expected total utility of the network under $\mathcal{P}$ is therefore:

$$\mu^{\mathcal{P}}(N) = \sum_{\xi \in \Omega^N} \mathbb{P}\{\xi\}\ U^{\mathcal{P}}(N,\xi) \tag{12}$$

We directly see the similarity with the definition of $r^{\mathcal{P}}(N)$ in (1). Additional utility measures such as the utility of the best static schedule (DSC utility variant), utility under perfect reoptimization (DDC variant), or even when provided an oracle that sees the future (DWC variant) can be obtained by simply replacing deterministic function $\Phi^{\mathcal{P}}(N,\xi)$ in (2), (3) and (6) by the deterministic utility function $U^{\mathcal{P}}(N,\xi)$.

Interruptible tasks and cutoff times. A classical assumption in the literature, which we made until here, is that failing at executing an activity automatically implies the failure of the entire network. However, in some contexts this assumption may not be appropriate. In fact, some activities may be considered as less critical, meaning that failing these activities does not interrupt the execution of the temporal network. Such activities could be interrupted, without preventing from executing subsequent activities, that is, without necessarily resulting in a global execution failure. In our rover application example, this could be true for any experimental activity, which are somehow isolated. Nonetheless, failing (interrupting) an activity may however turn impossible to carry out a related subset of remaining ones (*e.g.* an experiment composed of several tasks). In our example of Fig. 1, interrupting a driving activity would necessarily compromise the associated experiment, although it does not prevent from further relaying. On the other hand, in this example the relaying activities are critical, and should not be considered as interruptible. Note that a network composed only of interruptible activities would always be of robustness 1.

The fact that a task can be interrupted in case of a too long execution time implies the existence of a predefined cutoff time, which we assume to be u_{0j}, the upper bound of the time window of any time event t_j. Hence, the value assigned to any time event t_j will always be of at most $u_{0j}+1$, meaning that whatever happens, the execution of the network continues. Naturally, t_j can only be considered as successfully executed if assigned a value $\leq u_{0j}$ (and all other time constraints are fulfilled).

Limitations of the proposed computation method. Since $t_0 = 0$, cutoff defined relatively to t_0 actually is a deterministic absolute value (*e.g.* in Fig1, putting a cutoff at 600 time units for the t_{10} Rover 2 experiment time event). Another type of cutoff, which could be named relative cutoff, would reference to a maximum duration with respect to another time event (*e.g.* in Fig1, stating that t_{10} cannot last more than 100 time units from t_9). Although being a desirable operational property, unfortunately relative cufoff times are not supported by the computation method we describe in Section 5; only absolute

cutoff times are supported. Furthermore, in the context where a task is interrupted due to reaching its cutoff time, the question of subsequent task prerequisites arises. For instance, if *Rover1::drive* is interrupted and deemed failed, then *Rover1::expe* would not start execution, although *Rover1::relay* is not impacted. In this work, we assume that a task interruption due to cutoff time bound is not considered execution failure, *i.e.* subsequent tasks can still be executed. Cases where interruption are deemed execution failure are left for future work.

Controllability levels for interruptible tasks. It is important to note that the concept of interruptible tasks relies on alternative operational assumptions, thus applying on different operational contexts, in which the classical definitions of strong/dynamic/weak controllability are not valid. Alternative or extended definitions of these controllability levels to accommodate interruptible tasks are left for future work.

3.4 *NextFirst* dispatching protocol

The *NextFirst* protocol (Brooks et al., 2015), also known as *DC-dispatch* (Morris et al., 2001) or *early execution* (Lund, Dietrich, Chow, & Boerkoel, 2017), dynamically assigns a value to and dispatch each time point (*i.e.* executes the PSTN) in $\mathcal{O}(n)$ linear time, by starting activities as soon as possible. Following the definitions of Sec. 2.3, we then have $\Gamma_E^t \subset T_E^t$ in general, as at a current time t we are only interested in assigning a value to the very next executable time point(s), regardless subsequent ones (if any).

Let t_j be a controllable time point in a PSTN, and $I_j = \{(0,j), \dots, (i,j)\}$ the set of incoming edges in t_j. We assume t_j to be a controllable time point and I_j to contain controllable edges only, which one can easily enforce as shown in Fig. 4(c). Therefore, t_j is assigned a time value as soon as all the preconditions are validated, that is, all the $t_0, \dots, t_i$ time points are known, leading to the very simple online decision rule:

$$t_j = \max(t_0{+}l_{0j},\ \dots\ , t_i{+}l_{ij}). \tag{13}$$

In the case $t_j > \min(t_0{+}u_{0j},\ \dots\ , t_i{+}u_{ij})$, the dynamic execution is interrupted and considered as failed. Naturally, *NextFirst* protocol has linear complexity $\mathcal{O}(n)$. 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 $t_{12} \leq 700$.

The remaining of this paper heavily relies on the *NextFirst* dispatching protocol, for which the simplified operational assumptions allow an efficient computation of the DP-robustness as well as the DP-utility of a PSTN.

Suboptimality of *NextFirst*. The simplicity of *NextFirst* provides interesting computational properties, but comes at the expense of being suboptimal. Indeed, "*if we finish cooking too early, the dinner will be cold*" (Nilsson et al., 2014). As observed in Eq. (3), a more clever dispatching protocol assigns a time value based not only on past time events, but also in light of remaining the remaining uncertainty. Fig. 4(a) shows a basic PSTN for which operating t_1 as soon as possible, following (13), leads to a failure whereas postponing the execution of t_1 gives a valid schedule. In fact, in addition to be dynamically controllable, the STN at hand is also strongly controllable. Dispatching strategies such as DREA (Lund et al., 2017) or derivatives like DREAM (Abrahams, Chu, Diehl, Knittel,

Lin, Lloyd, Boerkoel Jr, & Jeremy, 2019) are capable of reaching a 100% success rate for this PSTN, but are considerably more complex, and do not give much insight into PSTN dynamic controllability. These strategies rely on creating fully decoupled "guides" over strongly controllable versions of the provided PSTN.

3.5 Complexity levels of dynamic controllability

In many operational contexts, under computationally limited settings (*e.g.* Mars Perseverance rover), a DP-robustness measure based on an appropriate suboptimal protocol may be more adequate than the DDC, as the latter relies on optimal online re-scheduling, which is intractable in practice. In other words, a PSTN which is theoretically DC may not be DC in practice. When activities are not interruptible, dynamic controllability is usually seen as an interesting, positive property for a PSTN. It means that no matter the uncertainty, it is always possible to succeed in executing the network, in theory, by adapting the decisions in a dynamic fashion. In practice however, it may be that the online decision system (*i.e.* dispatching protocol) involved requires tremendous computational resources, and therefore potentially significant (or even irrelevant) runtime. A more interesting property for a PSTN would then be to prove the network as being DC under some limited online computational resources. The current PSTN formalism lacks a theoretical tool that allows to characterize dynamically controllable (P)STNs, according to the tractability of the decision algorithm that actually enables DC.

The STN depicted in Fig. 4(a) is dynamic (and strongly) controllable, whereas it has the noticeable property of not being controllable by *NextFirst*. Fig. 4(b) depicts a non strongly controllable PSTN which is, unlike (a), dynamic controllable under *NextFirst*. Now, think of a PSTN that would be composed of both (a) and (b), *e.g.* by simply linking t_2 of (a) to t_0 of (b) with a $[0, \infty[$ constraint. The resulting PSTN would still be dynamic controllable, although too complex for *NextFirst*. We thus see that different decision complexity levels may be identified within a dynamically controllable PSTN. We propose two additional, complementary intermediate levels of dynamic controllability for PSTNs:

Definition 5 (linear dynamic controllability)**.** *A PSTN is said to be linearly dynamic controllable (linear DC) if a dispatching protocol of linear worst case time complexity suffices at reaching dynamic control.*

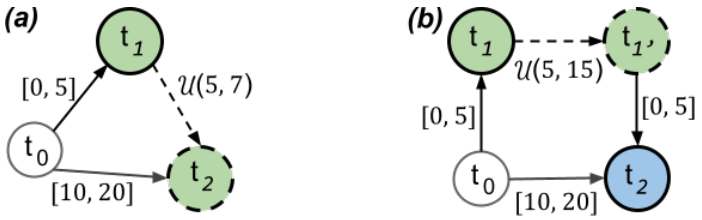

Figure 4: (a) Example of *NextFirst* suboptimality on a strongly/dynamically controllable network ($\mathcal{U}$ is for uniform). (b) Example of a network dynamically controllable under *NextFirst*, but not strongly controllable.

Definition 6 (polynomial dynamic controllability)**.** *A PSTN is said to be polynomialy dynamic controllable (polynomial DC) if a dispatching protocol of polynomial worst case time complexity suffices at reaching dynamic control.*

Definition 7 (hard dynamic controllability)**.** *A PSTN is said to be hardly dynamic controllable (hard DC) if there exists a dispatching protocol that reaches dynamic control on that network, but no polynomial-time algorithm that computes it.*

Strong controllability is a particular case of linear DC. Any strong controllable PSTN by definition enables dynamic control through a $\mathcal{O}(1)$ dispatching protocol. We may say it is constant time DC. Both (P)STNs of Fig. 4(a) and (b) are linear DC, even though (a) is not controllable by using *NextFirst*. Also, any PSTN having a DP-robustness of 1 under *NextFirst* is therefore further qualified as linear DC. On the contrary, in general a *NextFirst* DP-robustness lower than 1 does not necessarily imply hard DC, as a different dispatching protocol may still be able to achieve DC, such as PSTN (a) of Fig. 4. By combining PSTNs (a) and (b) of Fig. 4, we obtain a PSTN that is DC, although not by using *NextFirst* and, unlike (a) alone, not strongly controllable. On the other hand, a hardly DC network means that only a perfect (NP-hard) reoptimization approach may guarantee to succeed under dynamic control in any possible situation. In practice however, perfect reoptimization is usually not an option.

Historically, (Tsamardinos, Muscettola, & Morris, 1998) first studied the concept of *dispatchability* in the context of deterministic STNs, when all activity durations are known in advance and one is still seeking for some execution flexibility, by determining the schedule dynamically according to current execution conditions. They recognize the limitation of online computing, and therefore the necessity for fast efficient decision steps. They define a family of STN dispatching protocols, called a "Dispatching Execution Controller", which has the particularity of being computationally bounded to linear time complexity. In their framework, a STN is qualified as dispatchable if it is always correctly executed by such dispatcher. In other words, the concept of dispatchability defined for STNs is a particular case of linear DC: any dispatchable STN is linear DC, although composed of no contingent edges. In fact, they proved that any STN can be transformed, in polynomial time, into a dispatchable STN by the addition of a reasonable amount of time constraints.

In Sec. 5, we show how to compute the DP-robustness under the linear time dispatching protocol *NextFirst*, thereby also allowing to prove linear DC when the computed value is one. Similar proofs of linear (or even polynomial) DC may also be obtained by considering alternative dispatching protocols. However, such arguments are specific to a dispatching protocol, and are not sufficient to prove hard DC on a PSTN. Deciding the hardiness of a dynamically controllable network is therefore an open question, left for future investigation.

4. Landscape of DDC/DSC and Utility Measures

Table 1 summarizes contributions on computing (or approximating) the DSC or DDC of a network, in the case of classical PSTNs. We do not cover PSTN extensions such as PSTNs with choices (Conrad, Shah, & Williams, 2009), conditional PSTNs (Combi, Posenato, Viganò, & Zavatteri, 2019), PSTNs with resource usage (Kumar, Wang, Kumar, Rogers, & Knoblock, 2018). Recent studies, such as (Cui & Haslum, 2019), has focused on control-

	Comp. value			Prob. distribution			
Degree of Strong Controllability	Exact	LB	Approx.	∼U	∼N	All	Cont.
(Akmal et al., 2019)		✓	✓	✓			✓
(Santana et al., 2016)		✓	✓	✓	✓		
(Cui et al., 2015)			✓	✓			✓
(Tsamardinos, 2002)			✓	✓	✓	✓	
(Wang & Williams, 2015)	✓	✓	✓	✓	✓	✓	✓
Our proposal		✓	✓	✓	✓	✓	
Degree of Dynamic Controllability	Exact	LB	Approx.	∼U	∼N	All	Cont.
Monte Carlo			✓	✓	✓	✓	✓
(Akmal et al., 2019)			✓	✓			✓
(Cui et al., 2015)			✓	✓			✓
(Vaquero et al., 2019)			✓	✓	✓		✓
(Cesta, Oddi, & Smith, 1998)			✓	✓			
(Wilson et al., 2014)			✓	✓			
(Lund et al., 2017; Abrahams et al., 2019)		✓	✓	✓	✓	✓	✓
(Saint-Guillain, 2019)			✓	✓	✓	✓	
Our proposal		✓	✓	✓	✓	✓	

Table 1: DSC (top) and DDC (bottom) measurement methods. The key properties we consider are **1)** Quality of the *computed value*: exact DSC/DDC value, lower bound, approximation, and **2)** Properties of the supported *probability distributions*: Uniform (∼U), Normal (∼N), ordinary distribution (All), and whether distributions can be continuous (Cont.).

lability checking and computing dynamic decisions (or reliable policies) for dynamically controllable PSTNs, whereas we focus on the DSC or DDC of uncontrollable PSTNs.

Classical assumptions on PSTN random variables. The inherent complexity of dealing with interconnected random variables requires, when not relying on sampling based methods (such as Monte Carlo), to impose strong assumptions to the nature of the system at hand. Existing work either assume or are focused on parametric probability distributions, such as uniform or normal, in order to describe the uncertainty on the PSTN contingent edges. In fact, we provide the first method able to deal with ordinary distributions, by relying on closed form equations, that is, no simulations nor approximations. Moreover, all existing works (including the present work) assume independence between realizations of the contingent constraint random variables, except for (Fang et al., 2014), which however did not consider the DSC/DDC estimation problem, and for simulation based approaches which, however, come with no guarantees. Furthermore, most of the methods overestimate the DDC by naively considering each contingent edge separately, hence failing at capturing how delays propagate through the structure of the network. On the contrary, in this paper we present a computational method that literally follows the propagation of the uncertainty within this structure, which ensures, if not the exact DDC computation (under specific operational assumptions — i.e. *NextFirst*), to never overestimate.

Degree of strong controllability. The contributions in this front are summarized in the top portion of Table 1. Most of the existing approaches aim at either solving, or approximating, problem (2), that is, the problem of finding a static schedule maximizing its DP-robustness. (Santana et al., 2016) proposed a pseudo-polynomial time algorithm to compute strong policies to PSTNs which are normally not strongly controllable, by squeezing the probability distribution bounds. In addition to providing a schedule, they compute an upper bound on the risk involved at squeezing these distributions. Yet, their computational framework allows to consider any kind of probability density function (uniform, Gaussian, etc.) to describe the contingent durations, as long as it is unimodal and monotonic on both sides of the mode. In other words, in (Santana et al., 2016) a strong schedule is computed as soon as the probability bounds get squeezed enough to enable strong controllability. The reported risk, namely an upper bound on the probability of that schedule to fail in practice, that is subject to the real probability bounds, relates to the degree of strong controllability later introduced in (Akmal et al., 2019), for STNUs only. (Wang & Williams, 2015) proposed a (non-polynomial) method for determining optimal static schedule minimizing the risk. Thanks to the independence assumption, the schedule's DP-robustness is exactly computable.

The computational framework proposed in this paper may be used to compute the DP-robustness of any static schedule, without making any assumption on the shape of the probability distributions, as long as these can be considered independent and discrete.

Degree of dynamic controllability. In the context of dynamic controllability, (Akmal et al., 2019) propose an approximation technique, based on a linear programming formulation of the STNU (*i.e.* assuming uniform distributions), achieving good accuracy rate. In (Brooks et al., 2015), a Monte-Carlo sampling approach approximates robustness under *NextFirst* protocol (and therefore approximates the DDC). Other approximated robustness metrics have been considered: Cesta et al. (1998) and Wilson et al. (2014) see the DDC 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. Huang, Lloyd, Omar, and Boerkoel (2018) introduce additional flexibility measures, based on a representation of the STN (*i.e.* no contingent edge) as a polyhedron. All of these approaches suffer from the fact that they do not directly deal with uncertainty in durations.

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) defines 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. The method proposed in (Saint-Guillain, 2019) attempts at computing an exact lower bound on the DDC, while handling ordinary distributions. Yet, it applies to particular PSTNs only, leading to an approximation in the general case. In (Lund et al., 2017; Abrahams et al., 2019), one of the two proposed methods (called SREA) leverages the concept of strong controllability in order to infer a lower bound on the DDC. In fact, SREA attempts to find a lower bound on the DP-robustness for its given dispatch

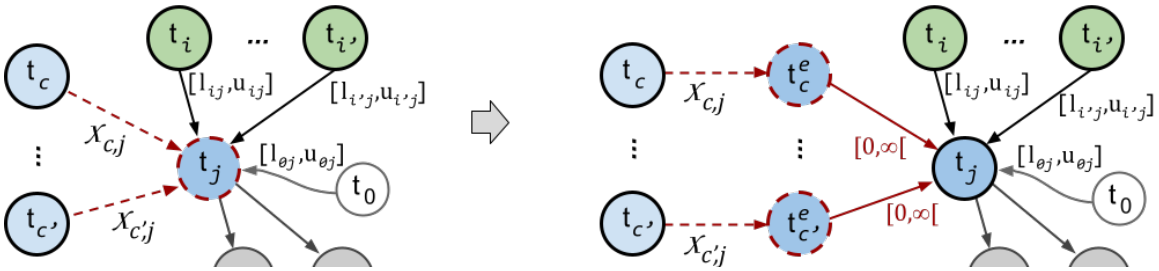

Figure 5: A network can be transformed in order to avoid synchronization points involving contingent incoming edge(s), which works as follows. For each synchronization point, every incoming contingent edge (with probability distribution $\mathcal{X}_{c,j}$) is replaced by a sequence: contingent edge (same random variable), new (contingent) time point, controllable edge $[0, \infty[$. Replacements are shown in red.

strategy, which appends to be *NextFirst*. The computed lower bound however decreases exponentially with the number of contingent edges, which makes it less informative in practical applications, as it rapidly becomes too small compared to the true DDC. More specifically, they compute a global, minimal acceptable risk level α, which is the allowable amount of probability mass to be sacrificed for all contingent edges, further leading to the lower bound $(1-\alpha)^{|C_C|}$. This is rather a side theoretical contribution of their method, which is of more general purpose, and these bounds were actually not experimentally computed.

In this paper, we show how the DP-robustness of a network can be computed exactly, thanks to operational assumptions simplifying the dynamic decisions, which provides an exact lower bound on the DDC.

Utility under strong and dynamic control. Up to our knowledge, there is currently no study related to PSTN expected utility measures, in particular when activities can in fact be safely interrupted. Interruption here should not be confused with that of conditional PSTNs, in which parts of the network could be disabled depending on realizations of conditional constraints, despite in this case, the expected utility measure would make sense as well. In this paper, we show how to compute an exact lower bound on the expected utility of a PSTN, when all the activities are interruptible, subject to predefined cutoff times.

5. Exact computation of DP-robustness and DP-utility

We now explain how the exact *NextFirst* DP-robustness (i.e. the expected probability of dynamic control success of an ordinary-distributed PSTN network when using *NextFirst* protocol) can be actually computed in fixed-parameter pseudo-polynomial time.

It is important to remark that, from now on, we make the assumption that our PSTN is composed of either uninterruptible tasks only, or interruptible tasks only. In fact, the probabilistic equations described below would not be valid anymore for a PSTN containing both interruptible and uninterruptible activities.

Assumptions and network preprocessing. We assume discretized time horizon and probability distributions. In practice however, continuous distributions may be used in input as long as their CDFs can be evaluated at discrete values. The horizon is noted $H = 1..h$. We also assume independence between the activity duration (*i.e.* contingent constraints T_C) 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 constraint is specified, we assume $[0, h]$. We call $t_{\mathrm{leaf}} \in T_C$ the *final time event* (*e.g.* t_{12} in Fig. 1). In case of multiple final events, we add a final synchronization point, that links these with $[0, \infty[$ edges.

Finally, following Fig. 5 we transform the network to avoid contingent synchronization points. The resulting network is equivalent under *NextFirst* assumptions, in the sense that given a particular scenario, the time value assignments are equivalent. In particular, the robustness under *NextFirst* remains unchanged. We prove that the replacement of a single particular contingent incoming edge keeps the time values assignments unchanged. Let the edge be (c, j). Under a scenario ξ, we denote d^ξ the realization of random variable $\mathcal{X}_{c,j}$, and following (13) then either **(a)** $t_j = t_c + d^\xi$, or **(b)** $t_j > t_c + d^\xi$. In case of **(b)**, the duration (c, j) in ξ is such that t_j's dispatching is waiting for another, contingent or controllable, time constraint. Consequently, replacing (c, j) as described in Fig. 5 cannot modify the time value of t_j in ξ, which remains consistent *w.r.t.* both t^e_c's value $(t_c + d^\xi)$ and the new $[0, \infty[$ edge. Note that in this case, we cannot specify $[0, 0]$ instead $[0, \infty[$, due to the multiple contingent edges arriving at the t_j. In case **(a)**, then in the transformed PSTN we have $t_j = t^e_c = t_c + d^\xi$ because of *NextFirst*, since by hypothesis $t_c + d^\xi$ realized as the maximum of t_j's incoming edges. Since the transformation of one unique contingent edge keeps the network equivalent (under *NextFirst*), problematic contingent edges can be transformed in turn until we obtain a PSTN without any contingent synchronization point.

Note that in general, it is also always possible to combine chains of uncontrollable events into a single uncontrollable events with a convolution. Similarly, one can always introduce more controllable events on any requirement triangle in a PSTN. It may be worth mentioning that this preprocessing step doesn't restrict the solution space in any way, and solutions to the processed form will lead to an equivalent solution in the original.

Walk-through example. This section will be supported by the simple PSTN case depicted in Fig. 6. For simplicity we will limit its application to the DP-robustness computation in the uninterruptible case. We have two synchronization points, t_u and t_s. The operational horizon is imposed to $[0, 20]$. There is an unique contingent duration from t_2 to t_u, uniformly distributed from 1 to 10 time units. We see that the main challenge here is imposed by the $[0, 2]$ time constraint from t_2 to t_s, and it is easy to see that the robustness of the network should be of 20%, has the execution would fail if the contingent duration realizes to a value higher than 2 time units. The network preprocessing step is also showed in Fig. 6, by adding the intermediate event t'_u, which prevents t_u from being a contingent synchronization point.

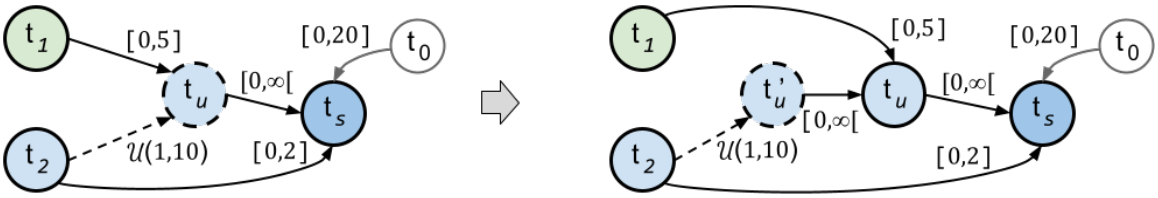

Figure 6: Walk-through PSTN example and its network preprocessing step.

5.1 DP-robustness.

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.

Once the network preprocessing step done, only the controllable time points are susceptible to cause execution failure. We are thus interested in the probability that every controllable time point gets assigned a time unit within its boundaries:

$$\begin{aligned} r^{\text{nf}}(N) = \ & \mathbb{P}\Big\{ \bigwedge_{t_j \in T_c} \max(t_0{+}l_{0j}, \ldots, t_i{+}l_{ij}) \leq t_j \leq \min(t_0{+}u_{0j}, \ldots, t_i{+}u_{ij}) \Big\} \\ = \ & \mathbb{P}\Big\{ \bigwedge_{t_j \in T_c} t_j \leq \min(t_0{+}u_{0j}, \ldots, t_i{+}u_{ij}) \Big\} \qquad (14) \\ = \ & \mathbb{P}\Big\{ t_{\text{leaf}} \leq \min(t_0{+}u_{0j}, \ldots, t_i{+}u_{ij}) \Big\} = \sum_{t \in H} P_{\text{leaf}}(t) \qquad (15) \end{aligned}$$

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}: (14) reduces to (15), 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 decision rule dispatching everything as soon as possible (13). $P_j(t)$ is recursively computed from t_j to t_0. This computation will be described in a moment. Note that, in general, the probability under *NextFirst* of assigning a valid time value to a time event t_j with time window $[l_{0j}, u_{0j}]$ is:

$$\mathbb{P}\{t_j \text{ succeeds}\} = \sum_{l_{0j} \leq t \leq u_{0j}} P_j(t). \qquad (16)$$

Considering our walk-through example, since t_2 starts at time 0 under *NextFirst*, we have

$$r^{\text{nf}}(N) = \ \mathbb{P}\Big\{ \ t_u \leq t_1 + 5 \ \wedge \ t_s \leq t_2 + 2 \ \Big\} = \ \mathbb{P}\Big\{ \ t_s \leq t_2 + 2 \ \Big\} = \sum_{t \in H} P_s(t)$$

where t_s is in fact our leaf time event. We will see soon that $P_s(t)$ equals zero for $t = 0$ and $t > 2$, so that we are left with $r^{\text{nf}}(a') = P_s(1) + P_s(2)$.

5.2 DP-utility.

An alternative but equivalent definition of $\mu^{\mathcal{P}}(N)$ in (12) can be devised by decomposing the computation over the time events. Let $w_j \geq 0$ be the utility value assigned to time event $t_j \in T$, and A_j random variable taking value 1 if t_e get successfully assigned a time

value by protocol $\mathcal{P}$, zero otherwise. Then,

$$\mu^{\mathcal{P}}(N) \;=\; \mathbb{E}\big[U^{\mathcal{P}}(N)\big] \;=\; \mathbb{E}\Big[\sum_{t_j\in T} w_j A_j\Big] \;=\; \sum_{t_j\in T} w_j\ \mathbb{P}\{t_j \text{ succeeds}\} \tag{17}$$

where, because $\sum_{t_j\in T} w_j A_j$ is linear, the expectation can be safely decomposed regardless the dependencies between the A_j's. Following (16), the DP-utility $\mu^{\mathcal{P}}(N)$ of N under *NextFirst* is then simply:

$$\mu^{\mathrm{nf}}(N) \;=\; \sum_{t_j\in T} w_j\ \mathbb{P}\{t_j \text{ succeeds}\} \;=\; \sum_{t_j\in T} w_j \sum_{l_{0j}\le t\le u_{0j}} P_j(t). \tag{18}$$

5.3 Time event probabilities.

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

$$f_j(t) \equiv \mathbb{P}\Big\{t = \max_{i:1..n}(t_i{+}l_i) \wedge\ t \le \min_{i:1..n}(t_i{+}u_i)\Big\} \tag{19}$$

be the probability that t_j may be assigned value t, when *not* considering its lower bounding constraint l_{0j}, if any. Time bounds $[l_{ij}, u_{ij}]$, $i : 0..n$ are noted $[l_i, u_i]$ for short. *NextFirst* protocol tells us t must be equal to $\max(t_1{+}l_1, \ldots, t_n{+}l_n)$, except if t comes too late, as suggested by the second condition of (19). Then, now considering constraints $l_0 = l_{0j}$:

$$P_j(t) = \begin{cases} \sum_{t'=0}^{l_0} f_j(t') & \text{if } t = l_0 \\ f_j(t) & \text{if } l_0 < t \le u_0, t \ge 0 \\ 1 - F_j(u_0) & \text{if } t = u_0 + 1 \text{ and is interruptible} \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

where $F_j(t)$ is the CDF of t_j: $F_j(t) = \sum_{t':1..t} P_j(t')$. The summation in the first case accounts for the situations in which (a) $t_i{+}l_{ij} < l_{0j}$ or (b) $t_i{+}l_{ij} = l_{0j}$. In (a), the system must wait until l_{0j} before executing t_j. The second case (b) accounts for situations in which the system can simply proceed without waiting. Finally, the third case only applies if t_j is interruptible (as well as all the other time events). The upper bound u_{0j} of the event's deterministic time window then acts as a *cutoff*, which ensures that the execution continues even in case of an interruption. It ensures to always assign a value to t_j, namely $u_{0j} + 1$ in case the dispatching protocol fails at assigning a valid value. Yet, this requires every event to be defined a consistent cutoff with respect to that of its predecessors, *i.e.*, if t_i is an ancestor of t_j, then $u_{0i} + l_{ij} \le u_{0j}$. The computation of $f_j(t)$ depends on the type of time point t_j it belongs to: either *transition* or *synchronization* 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:

$$f_j(t) = \sum_{0\le d\le t} p^d_{ij} \cdot P_i(t-d) \;\text{ if } l_{0j} - u_{ij} \le t \le u_{0j},\;\; 0 \text{ otherwise,} \tag{21}$$

where we see that if $f_j(t)$ could take positive values for $t < l_{0j}$, we must take upper bound time difference constraint u_{ij} in to account. In fact, in case the system gets ready for dispatching t_j at a time $t < l_{0j}$, meaning it must wait from t to l_{0j} to do so, it however cannot wait for more than u_{ij} time units. Back to our walk-through example, the transition points are t_1, t_2 and t'_u:

$$P_1(0) = f_1(0)1, \quad P_2(0) = f_2(0) = 1, \quad P_{u'}(1) = f_{u'}(1) = 0.1 \cdot P_2(1-1), \quad P_{u'}(2) = 0.1 \cdot P_2(2-2)$$

all other t values resulting in a zero probability.

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

$$f_j(t) = \mathbb{P}\Big\{ \underbrace{\bigwedge_{i:1..n} t_i{+}l_i \leq t \leq t_i{+}u_i}_{\alpha} \wedge \underbrace{\neg\big(\bigwedge_{i:1..n} t_i{+}l_i < t\big)}_{\beta} \Big\} \tag{22}$$

$$= \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i{+}l_i \leq t \leq t_i{+}u_i) \Big\} + \mathbb{P}\Big\{ \neg\big(\bigwedge_{i:1..n} t_i{+}l_i < t\big) \Big\}$$
$$- \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i{+}l_i \leq t \leq t_i{+}u_i \vee \neg\big(\bigwedge_{i:1..n} t_i{+}l_i < t\big) \Big\} \tag{23}$$

using the relation $\mathbb{P}\{\alpha \wedge \beta\} = \mathbb{P}\{\alpha\} + \mathbb{P}\{\beta\} - \mathbb{P}\{\alpha \vee \beta\}$. Here α and β together refer directly to the conjunction in (19). The $t \leq t_i + u_i$ terms ensure the second condition in (19), whereas the $t_i + l_i \leq t$ terms alone only impose $t \geq \max_{i:1..n}(t_i + l_i)$. The β part is there to strengthen to $t = \max_{i:1..n}(t_i + l_i)$, as t must be equal to at least one of the $t_i + l_i$'s.

Let us assume that all t_i's involved in the synchronization 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 (23) become straightforward to compute:

$$\mathbb{P}\{\alpha\} = \mathbb{P}\Big\{ \bigwedge_{i:1..n} t{-}u_i \leq t_i \leq t{-}l_i \Big\} = \prod_{i:1..n} F_i(t{-}l_i) - F_i(t{-}u_i{-}1) \tag{24}$$

$$\mathbb{P}\{\beta\} = 1 - \mathbb{P}\Big\{ \bigwedge_{i:1..n} t_i < t{-}l_i \Big\} = 1 - \prod_{i:1..n} F_i(t{-}l_i{-}1) \tag{25}$$

The synchronization points in our walk-through example of Fig. 6 are t_u and t_s:

$$\mathbb{P}\{\alpha_u\} = \big(F_1(t) - F_1(t-6)\big) \cdot \big(F_{u'}(t) - F_{u'}(t-1)\big) \qquad \mathbb{P}\{\beta_u\} = 1 - F_1(t-1) \cdot F_{u'}(t-1)$$
$$\mathbb{P}\{\alpha_s\} = F_u(t) - F_u(t-\infty) \cdot \big(F_2(t) - F_2(t-3)\big) \qquad \mathbb{P}\{\beta_s\} = 1 - F_u(t-1) \cdot F_2(t-1)$$
$$\mathbb{P}\{\alpha_s\} = F_u(t) \cdot \big(F_2(t) - F_2(t-3)\big)$$

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 $\mathbb{P}\{\alpha \vee \beta\} =$

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

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

$$\begin{aligned} 1{-}\mathbb{P}\Big\{ & (\boldsymbol{t_1 < t{-}u_1} \wedge \;\; t_2 < t{-}l_2 \;\; \wedge \ldots \wedge \; t_n < t{-}l_n \;\;) \\ & \vee (\; t_1 < t{-}l_1 \;\; \wedge \; \boldsymbol{t_2 < t{-}u_2} \wedge \ldots \wedge \; t_n < t{-}l_n \;\;) \\ & \ldots \\ & \vee (\; t_1 < t{-}l_1 \;\; \wedge \; t_2 < t{-}l_2 \;\; \wedge \ldots \wedge \; \boldsymbol{t_n < t{-}u_n}) \Big\} \end{aligned} \tag{26}$$

Let A_i be the random event in which the conjunction at line i of (26) 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:

$$\begin{aligned} \mathbb{P}\{\alpha \vee \beta\} &= 1 - \mathbb{P}\Big\{ \bigcup_{i:1..n} A_i \Big\} \\ &= 1 - \sum_{k:1..n} \left((-1)^{k-1} \sum_{\substack{I \subseteq \{1..n\} \\ |I|=k}} \mathbb{P}\Big\{ \bigcap_{i \in I} A_i \Big\} \right). \end{aligned} \tag{27}$$

Following our walk-through example,

$$\begin{aligned} \mathbb{P}\{\alpha_u \vee \beta_u\} &= 1 - \Big(\mathbb{P}\{A_1\} + \mathbb{P}\{A_{u'}\} - \mathbb{P}\{A_1 \cap A_{u'}\} \Big) \\ &= 1 - \Big(F_1(t-6) \cdot F_{u'}(t-1) + F_1(t-1) \cdot F_{u'}(t-1) - F_1(t-6) \cdot F_{u'}(t-1) \Big) \\ \mathbb{P}\{\alpha_s \vee \beta_s\} &= 1 - \Big(\mathbb{P}\{A_1\} + \mathbb{P}\{A_{u'}\} - \mathbb{P}\{A_1 \cap A_{u'}\} \Big) \\ &= 1 - \Big(F_u(t-\infty) \cdot F_2(t-1) + F_u(t-1) \cdot F_2(t-3) - F_u(t-\infty) \cdot F_2(t-3) \Big) \\ &= 1 - F_u(t-1) \cdot F_2(t-3) \end{aligned}$$

Since we assumed t_i variables to be mutually independent (not to be confounded with the assumed independence between contingent constraints T_C), then $\mathbb{P}\{ \bigcap_{i \in I} A_i \}$ 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 constraints have 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. Suppose one adds a constraint from t_3 to t_9, then a dependency appears between t_6 and t_{10}.

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:

$$f_j(t) = \sum_{t' \in H} f_j(t \mid t_a = t') \cdot P_a(t'). \tag{28}$$

In fact, $\{t_a = 0, t_a = 1, \ldots, t_a = h\}$ is a partition of Ω^N. 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.

In our walk-through example from Fig. 6, the synchronization points t_u and t_s are different cases. Since events t_1 and $t_{u'}$ share no common unpredictable ancestor, these can be considered as mutually independent, and we can compute the probabilities of t_u without imposing local independence:

$$\begin{aligned} f_u(t) = \;& \mathbb{P}\{\alpha_u\} + \mathbb{P}\{\beta_u\} - \mathbb{P}\{\alpha_u \vee \beta_u\} \\ = \;& \big(F_1(t) - F_1(t-6)\big) \cdot \big(F_{u'}(t) - F_{u'}(t-1)\big) + 1 - F_1(t-1) \cdot F_{u'}(t-1) \\ & - 1 + \Big(F_1(t-6) \cdot F_{u'}(t-1) + F_1(t-1) \cdot F_{u'}(t-1) - F_1(t-6) \cdot F_{u'}(t-1)\Big) \end{aligned}$$

In the case of t_s however, we see that t_2 and t_u appear as dependent. Their closest common ancestor is simply in fact t_2. Fortunately for both the author and the reader, t_2 is predictable as its only possible value is 0. Although counter-intuitive (how could t_u be independent of its ancestor t_s?), the independence easily appears if we acknowledge that, under *NextFirst*, an equivalent PSTN can be obtained by removing the $[0, 2]$ constraint from t_2 to t_s, and strengthening to $[0, 2]$ that from t_0 to t_s, as shown in Fig. 7. Consequently, t_s becomes a transition point, where by following Eq. (21):

$$P_s(t) = f_s(t) = P_u(t) \;\text{ if } t \leq 2, \quad 0 \text{ otherwise.}$$

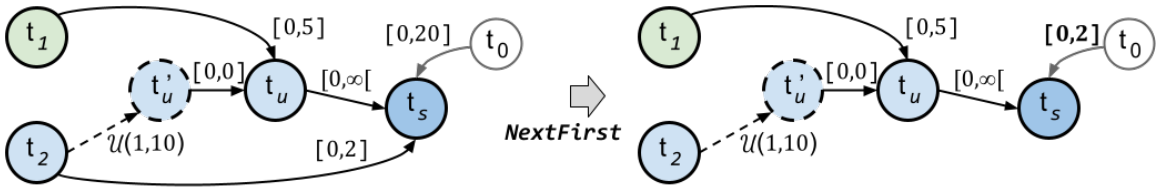

Figure 7: Modified *NextFirst*-equivalent walk-through PSTN example.

Back to (16), the robustness of the PSTN under *NextFirst* is therefore:

$$\begin{aligned}
r^{\mathrm{nf}}(N) &= P_s(0) + P_s(1) + P_s(2) = f_s(0) + f_s(1) + f_s(2) = P_u(0) + P_u(1) + P_u(2) \\
&= f_u(0) + f_u(1) + f_u(2) \\
&= F_1(0) \cdot \big(F_{u'}(0) - F_{u'}(-1)\big) + 1 - F_1(-1) \cdot F_{u'}(-1) - 1 + F_1(-1) \cdot F_{u'}(-1) \\
&\quad + F_1(1) \cdot \big(F_{u'}(1) - F_{u'}(0)\big) + 1 - F_1(0) \cdot F_{u'}(0) - 1 + F_1(0) \cdot F_{u'}(0) \\
&\quad + F_1(2) \cdot \big(F_{u'}(2) - F_{u'}(1)\big) + 1 - F_1(1) \cdot F_{u'}(1) - 1 + F_1(1) \cdot F_{u'}(1) \\
&= 1 \cdot (0 - 0) + 1 - 0 \cdot 0 - 1 + 0 \cdot 0 \\
&\quad + 1 \cdot (.1 - 0) + 1 - 1 \cdot 0 - 1 + 1 \cdot 0 \\
&\quad + 1 \cdot (.2 - .1) + 1 - 1 \cdot .1 - 1 + 1 \cdot .1 \\
&= .2
\end{aligned}$$

as $F_1(-5) = F_1(-4) = 0, F_1(0) = F_1(1) = F_1(2) = 1$, $F_{u'}(0) = 0$, $F_{u'}(1) = .1$, $F_{u'}(2) = .2$. Remark that even without turning t_s into a transition point, following the law of total probability in (28) still results in a 0.2 probability of success:

$$\begin{aligned}
f_s(t) &= P_s(0) + P_s(1) + P_s(2) = f_s(0 \mid t_2 = 0) \cdot P_2(0) + f_s(1 \mid t_2 = 0) \cdot P_2(0) + f_s(2 \mid t_2 = 0) \cdot P_2(0) \\
&= F_u(0) \cdot \big(F_2(0) - F_2(-3)\big) + 1 - F_u(-1) \cdot F_2(-1) - 1 + F_u(-1) \cdot F_2(-3) \\
&\quad + F_u(1) \cdot \big(F_2(1) - F_2(-2)\big) + 1 - F_u(0) \cdot F_2(0) - 1 + F_u(0) \cdot F_2(-2) \\
&\quad + F_u(2) \cdot \big(F_2(2) - F_2(-1)\big) + 1 - F_u(1) \cdot F_2(1) - 1 + F_u(1) \cdot F_2(-1) \\
&= 0 \cdot \big(1 - 0\big) + 1 - 0 \cdot 0 - 1 + 0 \cdot 0 \\
&\quad + .1 \cdot \big(1 - 0\big) + 1 - 0 \cdot 1 - 1 + 0 \cdot 0 \\
&\quad + .2 \cdot \big(1 - 0\big) + 1 - .1 \cdot 1 - 1 + .1 \cdot 0 \\
&= .2
\end{aligned}$$

since $F_u(t) = 0$ if $t < 1$, $F_u(1) = .1$, $F_u(2) = .2$, as we already computed $f_u(0) = 0$, $f_u(1) = .1$ and $f_u(2) = .1$. Finally, note that our calculations would have been even further simplified by noticing in Fig 7 that, t_1 being as predictable as t_2, event t_u could have been considered as a transition point as well.

Computational complexity. At synchronization time points, the complexity of computing (27), given t, depends on the maximum number $L \geq 1$ of incoming edges, from unpredictable events, at a synchronization point t_j. In fact, predictable events act exactly as the initial time event t_0, such as t_2 in Fig. 7, defining a deterministic time window for t_j. For example, t_1 and t_2 being predictable in our walk-through PSTN, we have $L = 1$, and $L = 2$ for our example of Fig. 1. At synchronization points, assuming the CDFs $F(\cdot)$ are incrementally maintained, (27) requires $\mathcal{O}(L2^L)$ operations, to be repeated at most h

times through (28). Following (20), $f_j(t)$ probabilities must be computed for each time unit t, and for each of the $m = |T|$ time events of the PSTN. Putting pieces together, the overall worst-case complexity of computing either r^{nf} or μ^{nf} is bounded by $\mathcal{O}(mh^2L2^L)$. It is equivalent to filling a matrix of size $m \times h$, each cell (j, t) being the probability $P_j(t)$ that event t_j gets assigned value t. If j is a transition point, then $P_j(t)$ requires at most h operations, based on the $(i, 1), \ldots, (i, h)$ cells of previous time point i, as described in (21). Otherwise, $P_j(t)$ hence requires the $\mathcal{O}(hL2^L)$ operations from (27) and (28). In brief: in case $L = 1$, overall complexity is $\mathcal{O}(mh)$. Otherwise, we have $\mathcal{O}(mh^2L2^L)$ in general.

DP-robustness of static schedules as lower bound on the DSC. The proposed method, designed for DP-robustness, can be directly generalized to the computation of DP-robustness $r^s(N)$ for any particular static schedule s. Following relation 8, it hence provides an exact lower bound on the network's DSC. Let $s : \ t_0 = 0, \ t_1 = t_1, \ \ldots, \ t_n = t_n$ a predefined static assignment for all the controllable time points of a temporal network. It is easy to see that the *NextFirst* dynamic protocol can be rendered static by simply imposing a time window $[t_i, t_i]$ on each controllable t_i. However, the problem of finding a good quality static schedule, leading to a high lower bound, is out of the scope of this paper.

Open source library. An open source C++ implementation of our method is available online, with usage guides and examples: https://bitbucket.org/mstguillain/dprobustness.

6. Experimental Validation and Applications

This section is an experimental validation, confronting the predicted values computed by the proposed computation method to averages measured on simulations (Sec. 6.1). Since a lower bound is actually only useful when it is reasonably close to the targeted value (*i.e.* the DDC), we hence compare the lower bound provided by *NextFirst* dispatching protocol with state-of-the-art approximations from the recent literature (Sec. 6.2).

We validate our approach on the same dataset as (Akmal et al., 2019),[1] involving 452 dynamically controllable and 110 uncontrollable instances, with uniform contingent durations. Durations are continuous, whereas our method requires a discretized 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.

1. These datasets are modified versions of ROVERS (for DC networks) and CAR-SHARING (for non-DC ones) datasets (Santana et al., 2016), downloaded from https://github.com/HEATlab/Prob-in-Ctrl

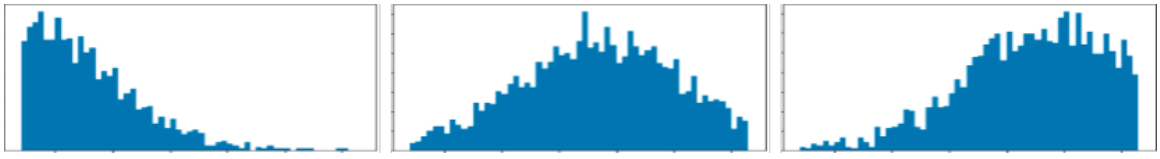

Figure 8: Examples of randomly generated ordinary distributions.

	min	max	mean	geo	$\lvert r^{\text{nf}}\text{-MC}\rvert$	$\lvert \mu^{\text{nf}}\text{-MC}\rvert$	$\Delta\max r$	$\Delta\max \mu$
$M = 10^2$	0.0003	0.011	0.0023	0.0017	0.027	0.026	0.11	0.13
$M = 10^3$	0.002	0.09	0.021	0.015	0.0085	0.0087	0.032	0.035
$M = 10^4$	0.03	0.87	0.20	0.14	0.0025	0.0025	0.009	0.009
$M = 10^5$	0.29	9.02	1.81	1.31	0.0009	0.0009	0.004	0.004
r^{nf} D=2	0.0002	3.95	0.34	0.09	*n.a.*	*n.a.*	*n.a.*	*n.a.*
r^{nf} D=3	0.0002	52.9	3.85	0.18	*n.a.*	*n.a.*	*n.a.*	*n.a.*
$M = 10^2$	0.0003	0.02	0.004	0.003	0.016	0.017	0.105	0.105
$M = 10^3$	0.002	0.21	0.044	0.027	0.0058	0.0061	0.034	0.049
$M = 10^4$	0.03	2.29	0.41	0.26	0.0017	0.0019	0.0144	0.0144
$M = 10^5$	0.25	23.75	3.84	2.35	0.0006	0.0006	0.0041	0.0041
$M = 10^2$	0.0005	0.03	0.006	0.004	*n.a.*	0.018	*n.a.*	0.11
$M = 10^3$	0.004	0.29	0.06	0.03	*n.a.*	0.0067	*n.a.*	0.041
$M = 10^4$	0.04	3.01	0.51	0.33	*n.a.*	0.0021	*n.a.*	0.009
$M = 10^5$	0.34	30.9	4.98	3.17	*n.a.*	0.0007	*n.a.*	0.004

Table 2: Comparison with Monte Carlo simulation: run time and prediction deviations. **Top:** non-interruptible tasks, uniform distributions. **Middle:** non-interruptible tasks, ordinary distributions. **Bottom:** interruptible tasks, ordinary distributions.

6.1 Computational method

We first consider the case when no task is interruptible. For each of the 110 uncontrollable PSTNs, we compare both the r^{nf} and μ^{nf} computed values with Monte Carlo (MC) simulations, which simply record the average success rate, and average achieved utility, of *NextFirst* protocol on M randomly sampled scenarios. Table 2 (top) gives an overview of the computation times (min, max, mean, geometric mean) as well as the average difference $|r^{\text{nf}} - \text{MC}|$ and $|\mu^{\text{nf}} - \text{MC}|$, and maximal difference $\Delta\max$ between predictions and simulations:

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. 8. Table 2 (middle) shows the average results obtained. The computational times using our computational method remain unchanged, whereas we notice an increased computation time for MC, due to more expensive sampling operations on non-parametric distributions.

Interruptible tasks. The latter experiment (*i.e.* using the ordinary distributions) is reiterated where we assume all tasks as being interruptible, this time by assessing the computed expected utility values only, as the robustness is now of 1 by definition. The results are listed in Table 2 (bottom).

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). Note that

the maximum number L of unpredictable events at a synchronization point, which critically impacts the computational complexity (in fact, it is the parameter of our *fixed-parameter* pseudo-polynomial time algorithm), never exceeds 2 in all the 110 uncontrollable networks, with 29 times events in average.

6.2 *NextFirst* dispatching protocol

We now compare our results, in terms of robustness values (DDC lower bounds), with the values computed by the two different state-of-the-art DDC approximation methods (referred to as Akmal^A and Akmal^B in what follows) proposed in (Akmal et al., 2019), on their own benchmarks. Naturally, just like Akmal et al. and for the remaining of this section, we assume that no task is interruptible. The time horizon has been rounded to D=3 decimals. We also computed the lower bounds obtained on the same PSTN instances by the SREA method (Lund et al., 2017), the only other lower bound computation method we found in the literature.

Set of uncontrollable PSTNs. We first address the benchmarks of 110 uncontrollable instances. In Table 3, each line gives the number of instances for which the DDC measure, as computed using either one of the four methods, is of at least the probability p. Compared to Akmal^A and Akmal^B methods, the average difference Akmal$^A - r^{\text{nf}}$ and Akmal$^B - r^{\text{nf}}$ are of respectively -0.011 and -0.029, that is a percent of success probability gain compared to Akmal^A and about 3% compared to Akmal^B . Recall that unlike r^{nf}, the DDC values obtained by Akmal^A and Akmal^B are approximations only. In other words, the computed bounds are as close to the true DDC than state-of-the-art approximations are, whereas unlike these approximations, r^{nf} is guaranteed to not overestimate the DDC. Considering the last column, which shows that 5 networks obtain a DDC of 1.0, it is thus important to distinguish between the conclusions of b and r^{nf}: contrary to method b which gives an indication that these 5 networks are likely to be DC, in the case of r^{nf} we then have valid proof of dynamic controllability. Finally, we observe that the lower bounds obtained with SREA method are significantly (around 60% in average) under those obtained by r^{nf}, with an average $\text{SREA} - r^{\text{nf}}$ difference of -0.595. This is not surprising, since SREA is designed to compute a lower bound not only on the DDC, but also on the robustness under *NextFirst* whereas our method computes the exact robustness for *NextFirst*. SREA attempts to capture probability mass from all contingent edges simultaneously, which severely weakens its ability to compute a tighter robustness bound (and thus its DDC lower bound).

Set of controllable PSTNs. When tested on the 452 dynamically controllable instances, *NextFirst* obtains 1.0 robustness on all of them. It means that. all these dynamically controllable instances are in fact linear DC, as defined in Sec. 3.5. They should clearly not be considered as "hardly controllable", since solved by such a simple dispatching protocol as *NextFirst*. Interestingly, the controllable set appears to be more challenging at a first glance than the uncontrollable set, as our L parameter here usually varies from 8 to 11, with 104 events in average. Despite this, even for these instances our algorithm always runs in less than a minute for each, on a macOS, Intel Core i7 2.3GHz, 16GB 3733MHz. Contrary to our method, SREA was unable to prove DC on any of these PSTNs.

p	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Akmal^A	110	97	92	86	80	73	67	61	49	32	0
Akmal^B	110	98	91	85	79	70	64	56	45	29	5
SREA	110	14	10	8	8	3	3	1	1	0	0
r^{nf}	**110**	**99**	**94**	**89**	**82**	**76**	**68**	59	48	**32**	**5**

Table 3: Comparison of the values computed by our method, and by state-of-the-art methods. For each line, the column indicates the proportion of PSTNs for which the corresponding a, b or r^{nf} method computed a robustness of at least 0.1, 0.2, *etc.*

Remarks. Our results show that despite its simplicity, *NextFirst* dispatching protocol is relevant, as it achieves noticeably good success rates. Compared with a state-of-the-art methods, the DP-robustness of *NextFirst* provides a pertinent, tight lower bound, on the true DDC of a PSTN. On the considered benchmarks, the obtained lower bounds reveal to fall closer to the DDC than state-of-the-art approximations are, even though the later are not guaranteed to not overestimate. Our method also significantly outperforms the only other method known for its ability to compute proven lower bound on the DDC. Finally, the experiments show that in practice, the *NextFirst* DP-robustness (and therefore our method) is also able to prove dynamic controllability in many cases, at least in all the DC networks considered in the later benchmark.

7. Applications: Mars 2020 Rover

Mars 2020 rover (also known as M2020 or Perseverance) is scheduled to land on Mars on February 2021 at the Jezero Crater. The rover aims to seek signs of ancient life and collect samples of rock for possible return to Earth.

Operating M2020 rover will involved frequent development of plans, in the form of task networks (including a set of science, engineering and communication activities), to be executed and elaborated onboard during a sol (i.e. a martian day). Such plan development process requires careful consideration of safety, resource and temporal constraints due to the uncertainty of the environment (Vaquero et al., 2019). In the latter, the traditional approach used in planetary rover missions is to add significant temporal margin for execution robustness; however, this can hamper rover efficiency (Gaines et al., 2016a). Ideally we would use task networks that not only are consistent and maximize the vehicle's productivity, but that are also robust to unexpected events and delays. The problem of evaluating robustness is therefore critical to handle temporal unpredictability.

In this section, we present two robustness analysis use cases in the context of M2020 rover, both based on our computational method.

7.1 M2020 Task Networks: Selected Sol Types and Scenarios

In the context of M2020 operations, a PSTN represents a task network, and is constructed based on a sol type and a corresponding stochastic knowledge, estimated here using data

Task Network	# Act	$\|T\|, \|T_E\|, \|T_C\|$	h.	$\|C\|, \|C_R\|, \|C_C\|$	sync μ, max	L
TN 0	32	67, 34, 33	92630	69, 36, 33	1.03, 10	2
TN 1	40	81, 41, 40	92630	90, 50, 40	1.11, 11	2
TN 2	28	57, 29, 28	92630	60, 32, 28	1.06, 12	2
TN 3	18	37, 19, 18	92630	39, 21, 18	1.06, 9	2
TN 4	42	85, 43, 42	92630	93, 51, 42	1.09, 10	2

Table 4: Studied M2020 sol types task networks with metrics of their the corresponding PSTNs: number of activities (# Act), time events (total, executable and contingent), time horizon size h, constraints, average (μ) and max. number of incoming edges at sync. point, and max. number L of unpredictable events at a sync point.

from previous Martian rover missions, specifically from Mars Science Laboratory (MSL) mission with Curiosity rover.

Sol types. Sol types represents the current best available data on expected M2020 rover operations during a sol (Jet Propulsion Laboratory, 2018). Each sol type represents the structure of a PSTNs: the time events and constraints. Five M2020 sol types, and thus PSTNs (or task networks) once added the activity duration probabilities, have been selected as candidates for our two robustness analysis. These networks are representative of what is currently being investigated and applied to develop an onboard scheduler for the M2020 rover (Chi et al., 2018; Agrawal et al., 2019). They generally contain between 20 and 50 activities, such as driving, conducting remote science, and taking images. In this study, all the activities have been assigned unary utility weight. Table 4 shows some of the main properties of each task network used in this work, including the number of activities, time events and constraints, as well as the size of the time horizon and metrics about synchronization points. Fig. 9 depicts a graphical representation of the structure of some of these networks, where each node stands for an activity. Blue activities are labelled as science tasks (e.g. SuperCam and MastCam imaging and scans), green ones are related to communication, and red ones stand for engineering tasks (e.g. rover drives).

Using MSL data. Since the M2020 rover is not yet in operations, accurate models of activity duration variance are not yet available. For the purpose of this study and to illustrate the benefits of the techniques, we use data and observations from previous Mars rover missions, specifically from Mars science Laboratory (MSL) rovers. Herein we use data from MSL operations covered across three case studies performed at JPL (Gaines et al., 2016a), representing a total of 65 sols, namely 65 observed scenarios for each network. We grouped the MSL rover activities into a few categories (e.g. engineering, science, comms) and study the activity duration variations during execution. We use this variations to inference the uncertainty on the M2020 rover activity durations and thus to obtain the PSTN's stochastic knowledge for this work. The problem of obtaining the each PSTN's stochastic knowledge, its probability distributions, is the subject of our first M2020 robustness analysis.

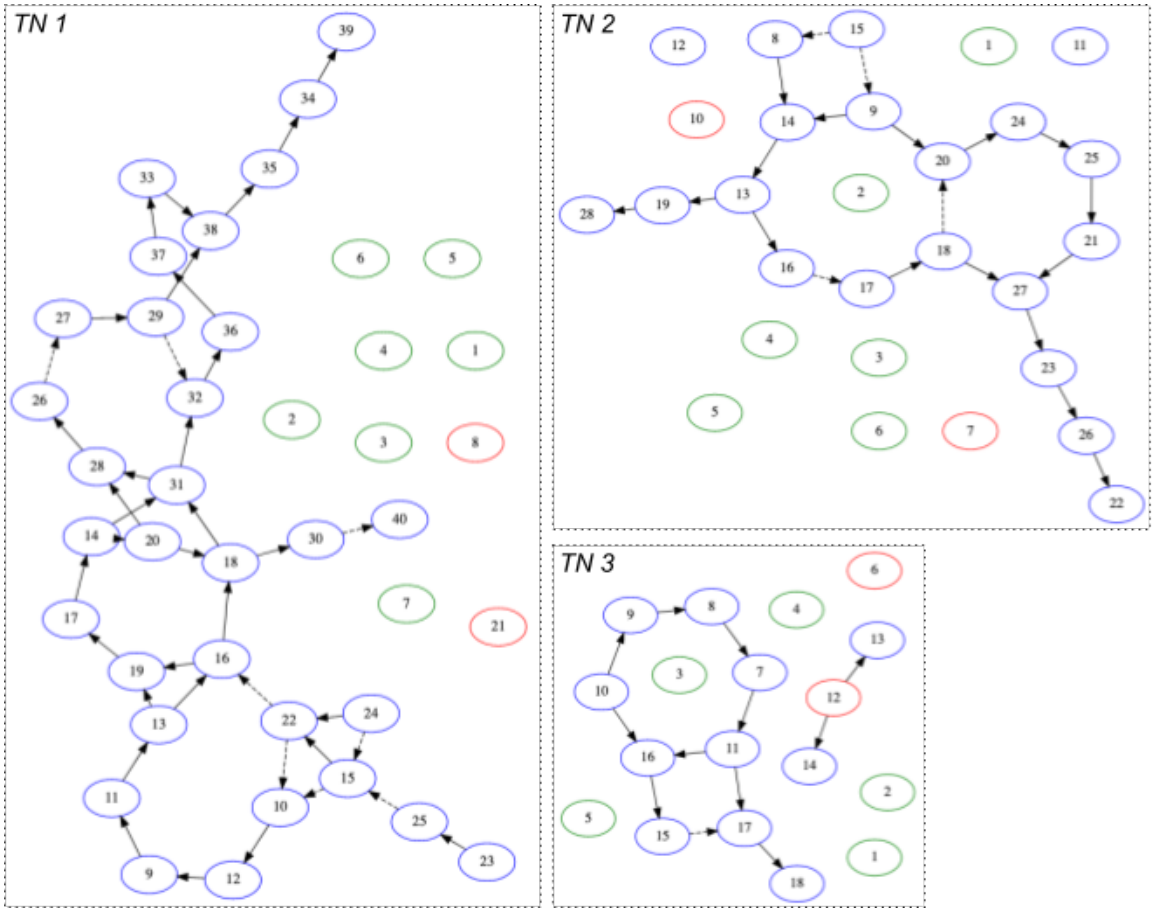

Figure 9: Sol types' activity dependencies. A circle is an activity, hiding a couple of starting/ending time events and a contingent constraint (the activity's duration). An arrow $a \to b$ represents a requirement constraint from either the start (dashed) or the end (bold) of activity a, to the start of b. Event t_0 is not shown, but is present and connected to each time even: every activity in the TNs has a finite time window.

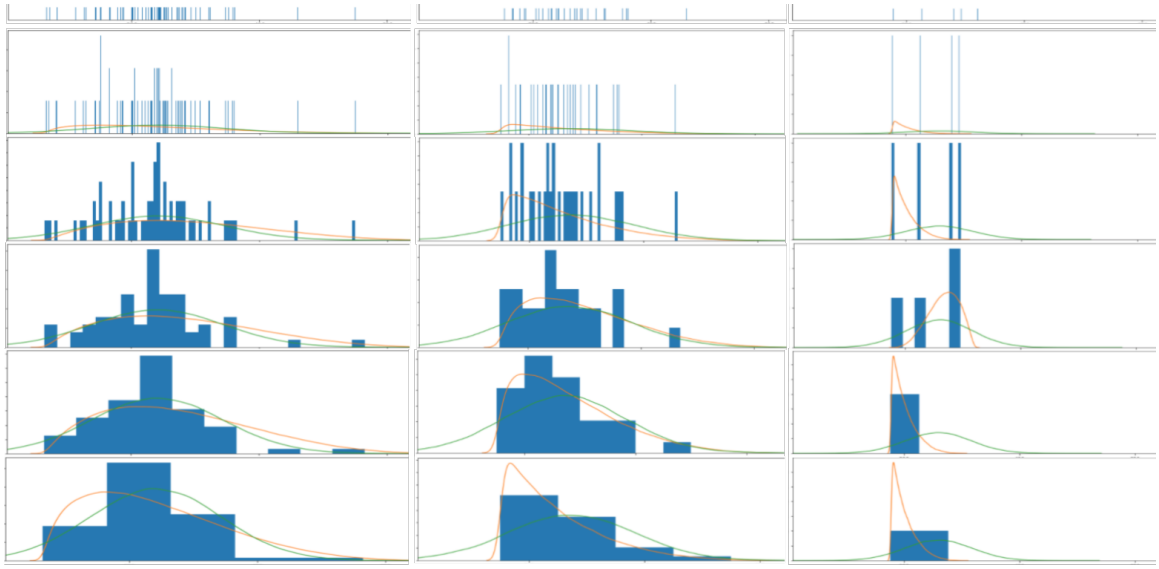

Figure 10: Ordinary (blue) and parametric (green: normal, orange: PERT) distributions obtained from empirical observations. The columns correspond to the amount of observations: 65, 30, 5. The first row shows raw distributions of data points. On subsequent rows, different distributions are obtained depending on how coarse grained discretization is applied, rounding to 1, 5, 20, 50, 100 time units.

7.2 Task Network Robustness Analysis - Comparing Probability Distributions

The MSL data generated in (Gaines et al., 2016a) has provided activity duration observations for a limited number of sols. Ideally, one would have a vast number of observations, but that is not the case here. The issue of only having a few observations, or even no observation at all, is also quite common in other application domains, such as human scheduling. In this application, we propose a preliminary analysis for determining the best representation of the available stochastic knowledge, in terms of probability distributions.

Consider an example activity, taken from a Mars 2020 task networks, for which we have 65 observations of its duration (one occurrence per sol). Namely, this activity has already been repeated 65 times, within conditions very similar to that of Mars 2020. Based on these observations, many different probability distributions may be obtained, as depicted in Figure 10. While a histogram obviously represents an ordinary distribution, parametric distribution such as PERT (orange in Fig. 10) or normal (green) can be obtained by computing the the histogram statistics: minimum, maximum and mode for PERT, mean and standard deviation for normal. In addition to PERT and normal parametric distributions, another widely used distribution for PSTNs is of course the uniform distribution. Note that the mode, and therefore the associated PERT distribution particularly, varies depending on how coarse grained discretization is applied to the histogram.

The question is then the following. Provided a limited number of observations, how to choose the best possible representation of our stochastic knowledge in order to best predict the robustness or utility of a given PSTN? Which discretization? Which probability

distribution? Depending on the number of observations, we empirically compare different distributions, based on the accuracy of the computed robustness and utility predictions.

Average results, obtained by considering the five different Mars 2020 PSTNs, are depicted in Fig. 11, assuming either uninterruptible tasks (top) and interruptible ones (bottom). The last 35 out of the 65 observed scenarios are kept for validation. That is, we assume that only (some of) the first 30 scenarios to be available for predicting our odds at succeeding (top of Fig. 11), or predicting the average utility (bottom), in the five PSTNs. Depending on the amount of observations (5, 10, 20, 30) considered, we report the accuracy of the predicted values w.r.t. to the averages as measured on the last 35 observed scenarios.

We first observe a somewhat very counter-intuitive result: the prediction accuracy decreases as the number of observations (from 5 to 30) increases. One would instead expect predictions based on more observations to be more accurate. This is certainly true *in the long term*. In our case study however, the accuracy can only be evaluated in the very short term, based on the last 35 observations (our validation set). This significantly limits the strength of the correlation between the observations and the validation set. From a statistical point of view, it makes the predictions more sensitive to *overfitting*, especially in the case of Uniform distributions. Uniform may give too much (equiprobable) importance to anything in between the two min/max observed values, whereas Normal distributions, for instance, are much less sensitive to outliers, hence providing a better protection against overfitting. We also remark that the results obtained on the robustness and the average utility are very similar. As observed in our next application in Sec. 7.3, in the case of the considered task networks, the average utility directly depends on the success probability of a couple of activities only. All time events being equally assigned a utility weight of 1, the predicted utility of a network containing 80 time events is always of at least 78.

Avoid Uniform distributions. At least for the considered PSTNs and our limited set of observations, both Normal and ordinary distributions tend to be a more adequate representation of our stochastic knowledge in general. Quite naturally, using Uniform distributions appears less interesting as the number of observations increases. Using PERT distributions seems to be less interesting than Normal or ordinary ones. Our analysis hence suggests to use either directly ordinary ones based on raw observations or Normal distributions when based on more than ten observed scenarios. Under that amount, Uniform distributions may be considered. Although ordinary distributions tend to behave slightly better than Normal ones, the difference is clearly not significant at the moment, and would require experimenting on given a higher number of observations. Our current results suffer from the lack of additional observations (especially our validation set which only composed of 35 scenarios). Nonetheless in practice, important decisions are to be taken when no more (or even less) knowledge is available; in that case, our recommendation is therefore to avoid Uniform distributions.

7.3 Task Network Structural Analysis

Our computational method allows to compute an exact lower bound on the probability of success of each time event of a PSTN, under dynamic control, following eq. (16). 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

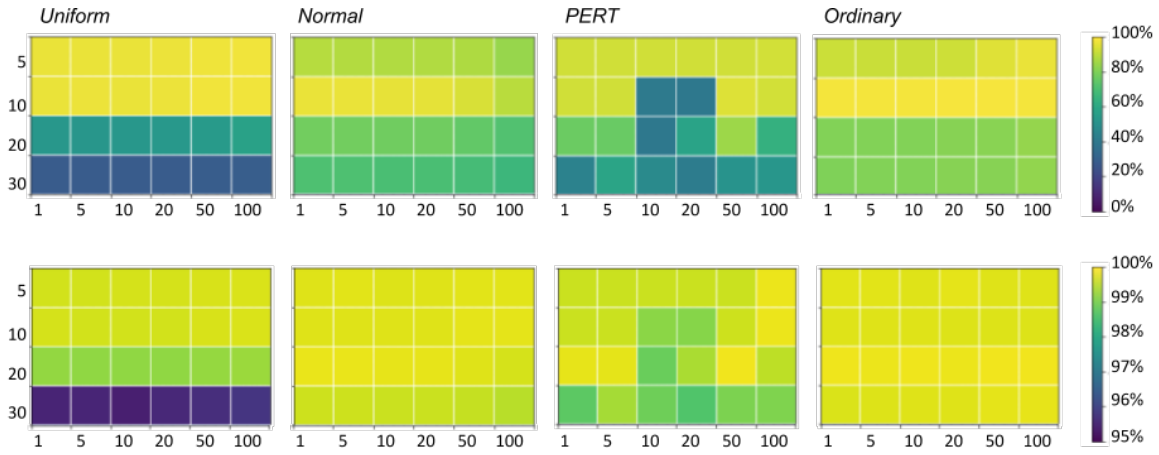

Figure 11: *NextFirst* robustness (top row) and utility (bottom row) prediction accuracy: from 0% (*resp.* 95%) to 100%, compared to the average success (*resp.* utility) rate measured on the remaining 35 observed sol scenarios. Rows correspond to the amount of observations: 5, 10, 20, 30. Columns correspond to how coarse grained discretization is applied, rounding to 1, 5, 10, 20, 50, 100 time units.

time events of the network, and on the network itself, of increasing (or decreasing) the uncertainty of an activity a by α%.

As a proof-of-concept, let us analyse a typical M2020 rover sol type represented as a PSTN of 18 contingent activities (36 time events). For the sake of simplicity, we use Normal distributions for our task networks. Increasing (or decreasing) an activity's uncertainty then simply amounts at modifying its standard deviation by α%.

Table 5 shows how the network gets structurally affected by $\alpha = +50$%. Cell at row i, column j, gives a amount of probability loss of activity j success when i's uncertainty is increased by 50%. Empty rows and column are not displayed. The second line of the table gives the initial success probability (%) of each activity; last columns r^{nf} and μ^{nf} stand for the entire network's DP-robustness (%) and expected utility. 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% robustness, and -0.29 expected utility. 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), however, now we can also explain how it propagates to other activities, and even map these conclusions with the correspond *TN 3* graph of Fig. 9.

Bottom of Table 5 shows results when all tasks are interruptible. The failure of activity 12 does not systematically entails the failure of subsequent activities anymore. In this context, 13 does not suffer anymore from an increase in the uncertainty of 12. Activity 14

	3	4	...	12	13	14	...	18	r^{nf}	μ^{nf}
	95.4	*51.0*	...	*90.5*	*90.5*	*90.4*	...	*100*	*43.7*	*17.2*
3	-8.5								-3.9	-0.08
4		-0.2							-0.2	-0.0
...			...							
12				-9.6	-9.6	-9.6			-4.7	-0.29
13					-0.0				-0.0	-0.0
14						-0.3			-0.1	-0.0
...							...			
18								-0.0	-0.0	-0.0

	3	4	...	12	13	14	...	18	μ^{nf}
	95.4	*50.7*	...	*90.5*	*100*	*98.4*	...	*100*	*17.3*
3	-8.5								-0.08
4		-0.2							-0.0
...			...						
12				-9.6		-1.4			-0.11
13					-0.0				-0.0
14						-1.2			-0.01
...							...		
18								-0.0	-0.0

Table 5: Structural brittleness analysis of M2020 task network *TN 3*. **Top:** non-interruptible tasks. **Bottom:** interruptible tasks

remains impacted by the uncertainty of 12, but not to the same degree (-1.4% instead of -9.6%). By being the only activity impacting the probability of success of at least one other activity, 12 is still considered here as the only structurally critical activity of the network.

Finally, Fig. 12 gives an overview on the structural dependency of another M2020 task network, composed of 40 activities. The analysis is now performed while considering $\alpha = -10\%$, that is, for each task the impact of a 10% diminution in its uncertainty. 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 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, even when activities are considered as interruptible (bottom of Fig. 12). These are typically the kind of activities that would not be spotted by previous brittleness analysis, unlike *e.g.* 5, 6, 21. Another observation is that there are actually a very few number of brittle activities in our task networks, and that each propagate to subsequent in a very limited scale, especially in the case of interruptible tasks. The same conclusion also appears in all our task networks, such as *TN 3* in Table 5. This could explains the obvious relationship between robustness and average utility in the results depicted in Fig. 11.

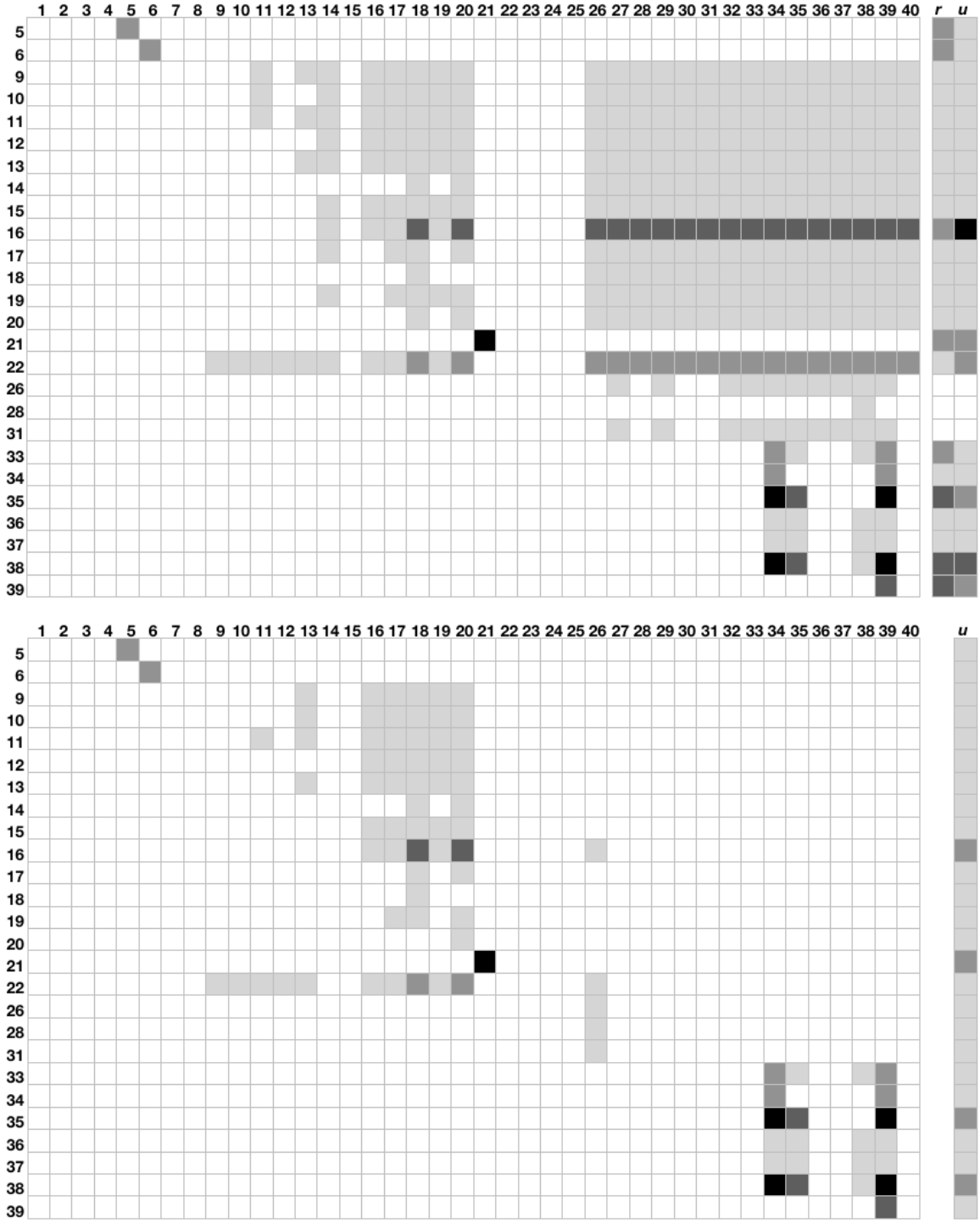

Figure 12: Structural dependency matrix of a M2020 task network of typical size: 40 activities. Up: no task interruptible. Down: all tasks interruptible. Empty rows are not shown. The darker the color of a cell, the bigger the impact.

8. Conclusions and Research Avenues

In (Saint-Guillain et al., 2020), 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 simplified operational assumptions, the computed robustness constitutes an exact lower bound on the degree of dynamic controllability under perfect dynamic assignment (DDC). 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, such as the proposed PSTN structural brittleness analysis. An open source C++ implementation of our method is available online, with usage guides, examples, as well as the benchmarks used in Section 6: https://bitbucket.org/mstguillain/dprobustness.

This paper brings significant additions to the theoretical contributions of (Saint-Guillain et al., 2020). We extend the concepts of DDC and DSC to PSTNs, introduce the degree of weak controllability (DWC) measure, and deduce remarkable fundamental inequalities from a mathematical analysis based on the proposed formal definitions. The aforementioned computation method is also extended in two ways. First, whereas the former paper only considered situations in which failing at executing one activity of the temporal network entails a global execution failure, in this paper we show how to deal with the opposite assumption, which is that activities may be interrupted by the use of a cutoff time, hence preventing from failing the entire execution. In such operational context, the robustness measure makes no sense anymore. We hence also extend our computation method to compute the so-called expected global utility of the network. All the experiments and application of our former study have been extended with results on expected network utility, and an additional application case is further described and applied to Mars 2020 rover task networks. In particular, preliminary results on a limited amount of observations discourage the use of Uniform probability distributions to describe activity durations in M2020 PSTNs.

We believe the developed method has great potential to be infused in Mars 2020 operations and rovers to come, as well as many other domains in operations research. In what follow we discuss some of these domains.

8.1 Other Potential Application Domains

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, Crama, & Pironet, 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

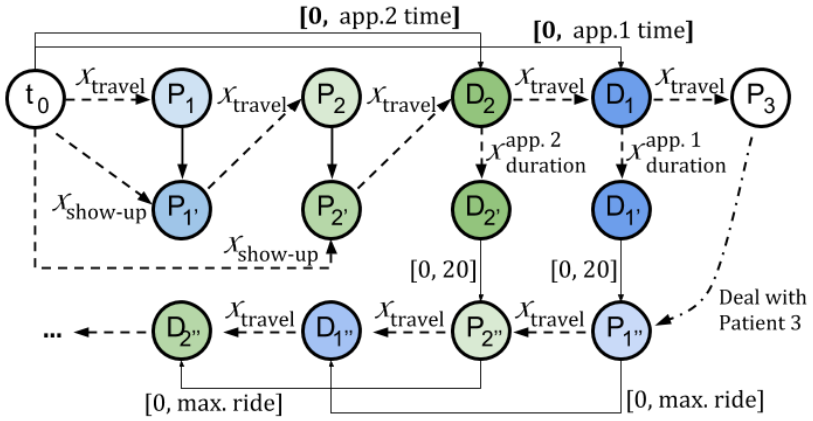

Figure 13: The vehicle first travels to Patient 1 (P1) location with a stochastic travel duration. The patient show-up time is also stochastic, and may be late. P1 is then picked up ($P_{1'}$) as soon as the vehicle arrives and the patient shows up. Then P2 is picked up ($P_{2'}$). Now P2 is dropped 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.

describing such a schedule, as illustrated in Fig. 13. 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. 14. 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.

Non-Temporal Dispatching. PSTNs encode constraints in terms of valid duration intervals, and unpredictable durations realizations, between events. In fact, it might be

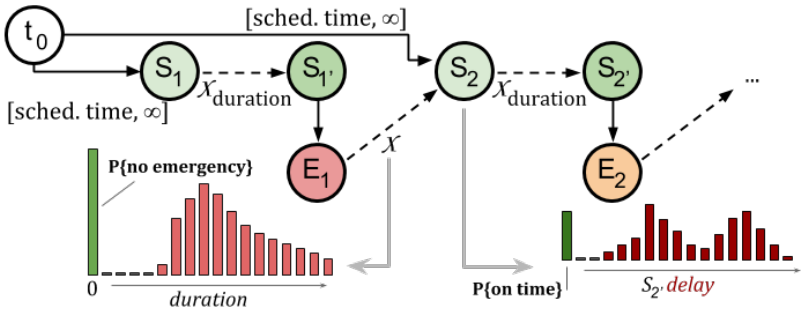

Figure 14: Operations start with scheduled surgery S_1, having uncertain duration. There is a probability that an emergency appears in the meanwhile. If it does, depending on its scheduled time, S_2 may be delayed. Our framework has the particularity to infer the time value assignment probabilities at any time event, such as S_2. This allows to compute expected delays, the probability to be on time, *etc.*

interesting to remark that the (P)STN formalism actually extends to not only temporal, but any possible notion of *distance* between events (*e.g.* space or quantity). For instance, spatial distances may equivalently replace temporal distances in a classical PSTN. Consider for instance the problem of piloting a complex aircraft braking system. In this context, suppose time is not an issue, and the dispatching decisions relate to where (and not when) to action which brake, knowing that some braking distances are uncertain and that, eventually, we must stop within the valid spatial interval of the airstrip. Furthermore, since the formalism only encodes distances, it is in fact agnostic to the nature of the encoded distances, and one could then encode PSTNs mixing scalar distances, such as both temporal and spatial constraints and uncertainties. In this case however, the most limiting aspect of the formalism in trying to mix several dimensions seems to be the controllable constraints, restricted in the STN formalism to *linear* binary constraints.

8.2 Future work and research avenues

We identified several promising research directions throughout the work described in this the paper.

Degree of dynamic controllability. Our theoretical results on the definitions of both the DDC (degree of dynamic controllability) and DWC (degree of weak controllability) show that not only an alternative lower bound on the DDC can be computed, but also that developing a method for computing the DWC would permit to bound the DDC from above. Up to our knowledge, there is so far no existing method able to provide such an upper bound on the DDC, which would provide valuable guarantees on the quality of any lower bound in terms of DDC closeness. Our multistage stochastic formulation of the DDC also suggests a new approximation method based on Monte Carlo Tree Search. MCTS has proven its efficiency at solving many different problems from various domains. As being a sampling based method, not only ordinary distributions can be easily handled, but also the dependencies between random variables. Furthermore, this approach significantly

differ from previous ones as being the first sampling-based method to perform a simulations without being restricted by a specific dispatching protocol, hence directly approximating the behavior of an optimal online scheduler.

Interruptible activities and dependencies. In this work, we just started to explore the possibilities offered by interruptible activities, such as the network expected utility metric. In order to maintain a practical computational complexity, our method is limited to the particular cases where either no task is interruptible, or all are. In some contexts, however, having both interruptible and non-interruptible activities would make perfect sense. Furthermore, it may also be that a subset of tasks could not be even started in case some conditions are not met, such as the successful completion of (a subset of) previous tasks in the dependency chain (*e.g.* if pre-heating activity fails than subsequent science activities that have specific thermal requirements won't be able to start). Because of the combinatorial aspect involved in the calculation of the conditional probabilities, sampling based methods seem to be the only viable approach for now.

Strong controllability. We saw that the proposed framework allows, for a given static policy, to compute its exact DP-robustness, *i.e.* probability of success under strong controllability. This suggests a potential contribution to the landscape of strong controllability. In fact, the DP-robustness decreases monotonically, in the direction of the network's edges, as the function is being computed. In other words, when there is no interruptible task, the (unconditional) success probability of one task is necessarily *at most* equal to that of its predecessors (*e.g.* in Fig. 1, *Rover1::drive* is a predecessor of *Rover1::expe*). The latter property can be efficiently exploited by embedding our DP-robustness function in a branch and bound optimization process, in order to compute optimally robust static policies for any PSTN (or STNU) being not purely strongly controllable by nature, thus computing the exact degree of strong controllability. In this context, a constraint programming approach may also be considered. This would provide an alternative to the method proposed by (Wang & Williams, 2015), for now the only method computing the exact DSC of a PSTN.

Disjunctive Temporal Networks. Up to our knowledge, there is no existing study in the literature on determining the DDC, or even robustness, of a disjunctive temporal network under uncertainty (DTNU). By using the monotonic property described in the previous paragraph, one could be able to embed our method in a decomposition algorithm, such as branch ad bound, able to compute the exact DP-robustness (under *NextFirst*) of a DTNU by solving an optimization problem on the DTNU's many possible interpretations. As our method deals with ordinary distributions, considering PDTNs would also be possible.

Acknowledgments

The research was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The thank the anonymous reviewers for their valuable comments, suggestions, and even corrections. We thank Maxime Parmentier for his valuable suggestions at the very beginning of this research.

References

Abrahams, J. R., Chu, D. A., Diehl, G., Knittel, M., Lin, J., Lloyd, W., Boerkoel Jr, J. C., & Jeremy, F. (2019). Dream: An algorithm for mitigating the overhead of robust rescheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 29, pp. 3–12.

Agrawal, J., Chi, W., Chien, S., Rabideau, G., Khun, S., & Gaines, D. (2019). Enabling Limited Resource-Bounded Disjunction in Scheduling. In *International Workshop on Planning and Scheduling for Space (IWPSS)*, pp. 7–15.

Akmal, S., Ammons, S., Li, H., & Jr, J. C. B. (2019). Quantifying Degrees of Controllability in Temporal Networks with Uncertainty. In *Proc. of the 29th International Conference on Automated Planning and Scheduling (ICAPS-19)*.

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

Bhargava, N., & Williams, B. C. (2019). Complexity bounds for the controllability of temporal networks with conditions, disjunctions, and uncertainty. *Artificial Intelligence*, *271*, 1–17.

Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming*.

Brooks, J., Reed, E., Gruver, A., & Jr, J. C. B. (2015). Robustness in Probabilistic Temporal Planning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3239–3246.

Cesta, A., Oddi, A., & Smith, S. F. (1998). Profile based algorithms to solve multiple capacitated metric scheduling problems. In *AIPS*, pp. 214–223.

Chi, W., Agrawal, J., Chien, S., Fosse, E., & Guduri, U. (2019). Optimizing Parameters for Uncertain Execution and Rescheduling Robustness. In *29th International Conference on Automated Planning and Scheduling (ICAPS)*.

Chi, W., Chien, S., Agrawal, J., Rabideau, G., Benowitz, E., Gaines, D., Fosse, E., Kuhn, S., & Biehl, J. (2018). Embedding a Scheduler in Execution for a Planetary Rover. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Cimatti, A., Micheli, A., & Roveri, M. (2016). Dynamic Controllability of Disjunctive Temporal Networks : Validation and Synthesis of Executable Strategies. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3116–3122.

Combi, C., Posenato, R., Viganò, L., & Zavatteri, M. (2019). Conditional simple temporal networks with uncertainty and resources. *Journal of Artificial Intelligence Research*, *64*, 931–985.

Conrad, P., Shah, J., & Williams, B. (2009). Flexible Execution of Plans with Choice. In *Nineteenth International Conference on Automated Planning and Scheduling*, pp. 74–81.

Cui, J., & Haslum, P. (2019). Dynamic controllability of controllable conditional temporal problems with uncertainty. *Journal of Artificial Intelligence Research*, *64*, 445–495.

Cui, J., Yu, P., Fang, C., Haslum, P., & Williams, B. C. (2015). Optimising Bounds in Simple Temporal Networks with Uncertainty under Dynamic Controllability Constraints.. In *19th International Conference on Automated Planning and Scheduling*, pp. 52–60.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial intelligence*, *49*(1-3), 61–95.

Fang, C., Yu, P., & Williams, B. C. (2014). Chance-constrained Probabilistic Simple temporal problems. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, Vol. 3, pp. 2264–2270.

Gaines, D., Doran, G., Justice, H., Rabideau, G., Schaffer, S., Verma, V., Wagstaff, K., Vasavada, V., Huffman, W., Anderson, R., Mackey, R., & Estlin, T. (2016a). Productivity challenges for mars rover operations: A case study of mars science laboratory operations.. Tech. rep., D-97908, Jet Propulsion Laboratory.

Gaines, D., Anderson, R., Doran, G., Huffman, W., Justice, H., Mackey, R., Rabideau, G., Vasavada, A., Verma, V., Estlin, T., Fesq, L., Ingham, M., Maimone, M., & Nesnas, I. (2016b). Productivity Challenges for Mars Rover Operations. In *International Conference on Automated Planning and Scheduling, Planning and Robotics Workshop (PlanRob)*.

Huang, A., Lloyd, L., Omar, M., & Boerkoel, J. (2018). New perspectives on flexibility in simple temporal planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 28.

Jet Propulsion Laboratory (2018). Mars 2020 rover mission. https://mars.nasa.gov/mars2020. Accessed: 2021-02-04.

Kumar, T. K. S., Wang, Z., Kumar, A., Rogers, C. M., & Knoblock, C. A. (2018). Load scheduling of simple temporal networks under dynamic resource pricing. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Morris, P. (2014). Dynamic controllability and dispatchability relationships. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pp. 464–479. Springer.

Morris, P., Muscettola, N., & Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In *International Joint Conference on Artificial Intelligence*, IJCAI'01, p. 494–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Morris, P. H., & Muscettola, N. (2005). Temporal dynamic controllability revisited. In *Aaai*, pp. 1193–1198.

Nilsson, M., Kvarnström, J., & Doherty, P. (2014). Incremental dynamic controllability in cubic worst-case time. In *2014 21st International Symposium on Temporal Representation and Reasoning*, pp. 17–26. IEEE.

Paquay, C., Crama, Y., & Pironet, T. (2020). Recovery management for a dial-a-ride system with real-time disruptions. *European Journal of Operational Research*, *280*, 953–969.

Rabideau, G., & Benowitz, E. (2017). Prototyping an Onboard Scheduler for the Mars 2020 Rover. *Proceedings of the International Workshop on Plaanning and Scheduling for Space, IWPSS*.

Saint-Guillain, M. (2019). Robust Operations Management on Mars. In *29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, Berkeley, CA, USA.

Saint-Guillain, M., Stegun Vaquero, T., Agrawal, J., & Chien, S. (2020). Robustness computation of dynamic controllability in probabilistic temporal networks with ordinary distributions. In Bessiere, C. (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4168–4175. International Joint Conferences on Artificial Intelligence Organization.

Santana, P., Vaquero, T., Toledo, C., Wang, A., Fang, C., & Williams, B. (2016). Paris: A polynomial-time, risk-sensitive scheduling algorithm for probabilistic simple temporal networks with uncertainty. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, pp. 267–275.

Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2014). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, A., Dentcheva, D., & Ruszczynski, A. P. (2009). *Lectures on stochastic programming: modeling and theory*, Vol. 9. SIAM.

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

Tsamardinos, I., Muscettola, N., & Morris, P. (1998). Fast transformation of temporal plans for efficient execution. In *AAAI/IAAI*, pp. 254–261.

Vaquero, T., Chien, S., Agrawal, J., Chi, W., & Huntsberger, T. (2019). Temporal Brittleness Analysis of Task Networks for Planetary Rovers. In *29th International Conference on Automated Planning and Scheduling (ICAPS'19)*.

Vidal, T., & Ghallab, M. (1996). Dealing with Uncertain Durations In Temporal Constraint Networks dedicated to Planning'. In *ECAI*, pp. 48–54. PITMAN.

Wang, A. J., & Williams, B. C. (2015). Chance-Constrained Scheduling via Conflict-Directed Risk Allocation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3620–3627.

Wilson, M., Klos, T., Witteveen, C., & Huisman, B. (2014). Flexibility and decoupling in Simple Temporal Networks. *Artificial Intelligence*, *214*, 26–44.