A more efficient and informed algorithm to check Weak Controllability of Simple Temporal Networks with Uncertainty

Ajdin Sumic[1] **Thierry Vidal**[1]

[1] Laboratoire Génie de Production - Université Technologique de Tarbes

asumic@enit.fr
tvidal@enit.fr

Résumé

Les réseaux temporels simples avec incertitude (STNU) sont un modèle bien connu basé sur les contraintes qui exprime des plans d'activités liées par des contraintes temporelles, chacune ayant des durées possibles sous la forme d'intervalles convexes. L'incertitude provient du fait que certaines de ces durées sont "contingentes", c'est-à-dire que l'agent qui exécute le plan ne peut pas décider de la durée réelle au moment de l'exécution. Pour vérifier que l'exécution satisfera toutes les contraintes, il existe trois niveaux de *contrôlabilité*, selon que l'agent observe ou non la durée réelle et à quel moment ; deux d'entre eux, la contrôlabilité forte et dynamique (SC/DC), se sont révélés à la fois utiles dans la pratique et prouvables en un temps polynomial. La troisième, la contrôlabilité faible (WC), suppose que les durées incertaines seront fixées par un "oracle" juste avant le début de l'exécution, et on suppose qu'elle est co-NP-complète. En outre, les algorithmes de vérification de la contrôlabilité sont des stratégies de propagation, qui présentent l'inconvénient habituel, en cas d'échec, de s'avérer incapables de localiser clairement les contingents qui expliquent la non-contrôlabilité. Cet article a trois contributions : (1) il justifie l'utilité de la WC dans les systèmes multi-agents (SMA) où un autre agent contrôle un contingent, et les agents se mettent d'accord juste avant l'exécution sur les durées ; il fournit un nouvel algorithme de vérification de la WC (2) dont la performance en pratique dépend de la structure du réseau, et semble être pseudo-polynomiale dans les réseaux faiblement connectés correspondant à des cas réalistes, et (3) qui fournit les cycles défaillants dans le réseau qui expliquent la non-WC : cela rendra donc possible dans les SMA de réparer les contingents identifiés par des négociations avec d'autres agents.

Abstract

Simple Temporal Networks with Uncertainty (STNU) are a well-known constraint-based model expressing plans of activities related by temporal constraints, each having possible durations in the form of convex intervals. Uncertainty comes from some of these durations being *contingent*, i.e., the agent executing the plan cannot decide the actual duration at execution time. To check that execution will satisfy all the constraints, three levels of *controllability* exist, depending on if and when the agent observes the actual duration; two of them, the Strong and Dynamic Controllability (SC/DC), have proven to be both useful in practice and provable in polynomial time. The third one, the Weak Controllabiilty (WC) assumes that uncertain durations will be set through some 'oracle' just before execution starts, and is conjectured to be co-NP-complete. Moreover, controllability checking algorithms are propagation strategies, which have the usual drawback, in case of failure, to prove unable to clearly locate the contingents that explain the non-controllability. This paper has three contributions: (1) it substantiates the usefulness of WC in multi-agent systems (MAS) where another agent controls a contingent, and agents agree just before execution on the durations; it provides a new WC-checking algorithm (2) whose performance in practice depends on the network structure, and seems to be pseudo-polynomial in loosely connected ones corresponding to realistic cases, and (3) which provides the failing cycles in the network that explain non-WC: that shall hence make it possible in MAS to repair the identified contingents through negotiations with other agents.

Keywords: Temporal constraints, Temporal uncertainty, Multi-agent planning, Graph-based algorithms, Explainable inconsistency

1 Introduction

Temporal Constraint Satisfaction Problems (TCSP) are constraint-based problem formulations that allow to represent and reason on temporal constraints. They are used

in a lot of domains, such as planning and scheduling (on which we will focus), supervision of dynamic systems, or workflow design. They are based on a graphical model, the reason why they are usually called Temporal Constraint Networks (TCN)[6] : variables/nodes are time-points for which one shall assign a timestamp. Constraints/edges express sets of possible durations relating them. A key issue is the ability to check the consistency of the whole network. The simplest class, called the Simple Temporal Network (STN), arises when they have only binary constraints with only convex intervals of values (no disjunctions). One of the main strengths of this restricted, but often sufficient in practice, model is that consistency checking is made through a polynomial propagation algorithm (the Floyd-Warhsall reduction) and provides a complete *minimal* network in which all inconsistent values are removed. This minimal network can be passed on to some execution manager that can take any value on the domain of the first activity to schedule, and repropagate, and so on iteratively.

A well-known extension of STNs that handles uncertainties, called STNU (Simple Temporal Network with Uncertainty), has been proposed by [13]. An STNU contains uncertain (*contingent*) durations between time-points which means the effective duration is not under the control of the agent executing the plan, which is useful for addressing realistic dynamic and stochastic domains.

In STNUs, the notion of temporal consistency has been redefined in the form of *controllability* : an STNU is controllable if there exists a strategy for executing the schedule whatever the values taken by the contingent durations. In [13] the authors introduce three levels of controllability that express how and when the uncertainties are resolved : the Weak Controllability (WC) proves a solution exists for any possible combination of contingent values. Which requires that some 'oracle' provides those values before the timing of controllable time-points is decided ; the Dynamic Controllability (DC) is more demanding as it assumes that at execution time a strategy can be built based on past observations only, thus whatever the contingent durations still to be observed ; last, the Strong Controllability (SC) is even more demanding as it enforces that there is one unique assignment of controllable timepoints values, which defines a static control strategy that works whatever the contingent durations will be at execution time. WC has often appeared to be unrealistic in dynamic applications that assume full progressive observability at execution time. DC answers to that and looks more relevant, and has received much attention in previous works. Comparatively, SC is usually too demanding unless under partial or non-observability, or when some strict commitment must be made on the execution schedule timing for some client.

Previous works prove that SC and DC can be resolved with specially designed propagation-based algorithms that run in polynomial time [10, 3, 13]. WC is more complex because it is supposed to be a co-NP-complete problem, and only exponential algorithms exist to check WC [5, 13]. This is another reason why WC has not received as much attention as DC and has been disregarded [2, 13].

Anyway, nowadays, STNUs appear to be useful in even more new application domains in which WC would actually be relevant, especially when it comes to multi-agent task management. This paper aims to address it, discussing more precisely how and when one could need it. As we will argue some of those applications definitely need more efficient algorithms : this paper aims at providing a new algorithm for checking WC in STNUs that is much more efficient in practice and even experimentally pseudo-polynomial under some conditions,i.e., behaves like a polynomial time algorithm. Contrary to the complete propagation algorithms proposed for SC and DC, our algorithm maintains and reasons only on the input constraints, which form paths in the networks. As in any graph, such paths join and form cycles. We prove that it is possible to check the global weak controllability by locally checking the elementary cycles of an STNU.

Moreover, the algorithm is also capable of diagnosing the source of uncontrollability of a non-WC STNU, i.e., to detect the set of constraints that makes the STNU not weakly controllable. This explainable issue is a problem recently addressed in STNU for DC in [9]. This is important for the executor manager in order to repair the schedule [2, 1, 12]. The paper is organized as follows : Section 2 first recalls the necessary background on STNU. Section 3 then discusses the usefulness in practical applications of WC. Then, we prove in Section 4 how local controllability on cycles is equivalent to global WC. Next, Section 5 will present how to locally check WC, and Section 6 will present the new algorithm for globally checking WC. Some experimental evaluation will be displayed in Section 7 before concluding our contribution with some prospects.

2 Background

2.1 STNU

A Simple Temporal Network (*STN*)[6] is a pair, (V, E), where V is a set of time-points v_i representing event occurrence times, and E a set of temporal constraints between these time-points, in the form of convex intervals of possible durations. A reference time-point v_0 is usually added to V which is the 'origin of time', depending on the application (might be e.g. the current day at 0 :00). The goal is to assign values to time-points such that all constraints are satisfied, which is equivalent to assigning a value to each constraint in its interval domain.

An STN with Uncertainty (*STNU*) is an extension in which one distinguishes a subset of constraints whose values are parameters that cannot be assigned but will be

observed. Before defining the model and the controllability levels, we introduce the usual basic notations :

— minimal bounds $l_{ij} \in \mathbb{R} \cup \{-\infty\}$,

— maximal bounds $u_{ij} \in \mathbb{R} \cup \{+\infty\}$,

— $\prec$ is the usual qualitative precedence relation between time-points : $v_i \prec v_j$, i.e., v_i happens before v_j. A numerical constraint in the STNU from v_i to v_j has its lower bound $l_{ij} \geq 0$ iff $v_i \preceq v_j$.

Definition 1 *(**STNU**) An STNU is a tuple (V, E, C) with :*

— *V a set of time-points $\{v_0, v_1, \ldots, v_n\}$, partitioned into controllable (V_c) and uncontrollable (V_u);*

— *v_0 the reference time-point : $\forall i, v_0 \preceq v_i$*

— *E a set of requirement constraints $\{e_1, \ldots, e_{|E|}\}$, where each e_k is of the form $[l_{ij}, u_{ij}]$ with, $v_i, v_j \in V$.*

— *C a set of contingent constraints $\{c_1, \ldots, c_{|C|}\}$, where each c_k is of the form $[l_{ij}, u_{ij}]$ with, $v_i \in V_c$, $v_j \in V_u$, and necessarily $v_i \preceq v_j : 0 \leq l_{ij} \leq u_{ij}$.*

Intuitively, controllable time points ($\mathcal{V}_c$) are moments in time to be decided by the scheduling agent, which is trying to satisfy all the requirement constraints (E) under any possible instantiation of the contingent constraints (C). Moreover, it's semantically impossible to have a contingent duration between two unordered time-points. Figure 1a is the graphical representation of an STNU.

In addition, an STN (and hence an STNU too) has an equivalent *distance graph* representation [6, 7]. Each constraint of the form $[l, u]$ between v_i and v_j would be represented as $v_i \xrightarrow{[l,u]} v_j$ in the STN, or equivalently through two corresponding edges in its distance graph : $v_i \xrightarrow{u} v_j$ and $v_j \xrightarrow{-l} v_i$.

2.2 Three levels of controllability

In a STN, the notion of consistency is applied to define that there exists a solution that satisfies the constraints. In a STNU, the consistency has been redefined in three levels of controllability that we are going to recall before focusing on the problem of Weak Controllability.

Definition 2 *(**Schedule**) A schedule δ of an STNU $\mathcal{X}$ is an assignment of the controllable time-points $\delta = \{\delta(v) \mid v \in V_c\}$*

Definition 3 *(**Situation and Projection**) Given an STNU $\mathcal{X}$, for all $k = 1 \ldots |C|, c_k = [L_k, U_k], \Omega = [L_1, U_1] \bigtimes \ldots \bigtimes [L_{|C|}, U_{|C|}]$ is the domain of all possible **situations** of $\mathcal{X}$.*
*A tuple $\omega = \langle \omega_1 \in [L_1, U_1], \ldots, \omega_C \in [L_{|C|}, U_{|C|}] \rangle \in \Omega$ is called a complete situation of $\mathcal{X}$ and $\mathcal{X}_\omega$ the **projection** of $\mathcal{X}$, is an STN where $\mathcal{X}_\omega = (V, E \cup C')$ with $C' = \{[\omega_k, \omega_k]\} \mid c_k \in C$*
*Last, a schedule δ_ω which satisfies all the constraints in $\mathcal{X}_\omega$ is called a **solution** of $\mathcal{X}_\omega$.*

Intuitively, the projection X_ω is a problem without uncertainty in which each contingent duration has been fixed to a given value. Hence X_ω is an STN.

Definition 4 *(**Weak Controllability (WC)**) An STNU $\mathcal{X}$ is **weakly controllable** iff $\forall \omega \in \Omega, \exists \delta$ such that δ is a solution of $\mathcal{X}_\omega$.*

This definition implies that the scheduler is clairvoyant, i.e. there is an 'oracle' that predicts the future and communicates the durations of the contingents to the scheduler before execution time. In other words WC requires all projections to be consistent independently from one another.

The two other controllablity levels impose more requirements, first (DC) removing the clairvoyance assumption and demanding that each assignment of a controllable time-point only depends on the past observations and not the future ones, and even (SC) demanding that the (unique) schedule is totally independent from any observation [13].

As said before, propagation-based checking algorithms exist for SC and DC [13][10][3]. But not for WC checking, which is conjectured to be co-NP-complete ; anyway, considering only the bounds is enough to verify any level of controllability in STNU : *it is enough to consider projections on the bounds to be sure that all intermediate projections will be consistent as well*[13]. The original algorithm to check WC checks the consistency of all $2^{|C|}$ STNs obtained by replacing the contingents with one of their bounds (upper or lower), which is an exponential algorithm.

3 Weak Controllability

In this section, we will argue that WC may be more relevant than DC and SC for some applications, while checking WC has been somehow left apart in the literature so far. This is mainly because WC assumes that the duration of the contingent activities is revealed when execution starts. Somehow, it requires an entity capable of predicting and sharing these durations with the scheduler agent before the actual occurrence of such activities.

However, in classical planning and scheduling applications, uncertainties come from external causes ; they are somehow 'controlled by Nature', and can only be observed at their time of occurrence. Which is why DC/SC are more relevant in practice. For instance, when one starts cooking a steak, they do not know how long it will take and will observe it once the steak is actually well done.

However, in many domains (logistics, transport, services), they have a first strategic phase that builds a plan without assigning all real resources ; a more precise tactical version will do that later. For instance, in a health service

or in a construction site, one needs a weekly plan for visiting patient rooms, or for the construction tasks, but the assigned teams (number of people, skills) are not known, which results in flexible and large enough intervals of possible durations. The precise assignment is only known each day for the next day, which allows to get a more precise plan just before execution, which is exactly the definition of WC.

Moreover, uncontrollable durations also appear in multi-agent systems, when some event might be owned by another agent instead of Nature. Still, in such contexts, collaboration may rely on the timely communication of effective durations at execution time. But in some application contexts, it might be the case that the interval of activity's duration represents the degree of freedom, and hence the *flexibility*, that some agent wishes to keep as long as possible to be more robust. But at the same time, actual durations must be set and communicated just before execution to the other agents that depend on it for better coordination. Thus, some tasks might be controllable (requirement) for one agent but uncontrollable for another one (contingent). That may arise, for instance, in collaborating hospital services that share common resources : they plan in advance their weekly operations with maximum flexibility but must set and confirm to others their own schedules each day for the next one. At this point, all shared events that are not controlled by Nature become requirement constraints at execution time. In a setting where no events are controlled by Nature, checking WC instead of DC/SC enables the agents to be more robust through least-commitment strategies, retaining flexibility as long as possible and hence computing more optimal solutions for their schedule.

One should notice that even though some approaches exist to deal with multi-agent STNUs [4], those actually consider a global plan shared among agents in which contingent constraints are still controlled by Nature. Considering inter-dependent STNUs, with shared activities that are controllable by one agent and only observed by others, still need to be investigated. That is the topic of future research for which this paper somehow is a cornerstone.

4 From local controllability to global controlability

4.1 Updated STNU graphical model

A starting point for resolving the issue of Weak Controllability is to add some features to STNU's graphical representation and adapt the model accordingly. Nodes in an STNU will not only be divided between controllable and uncontrollable time-points (the ending time-points of some contingent constraint), but we also need to identify among them *divergent* time-points and *convergent* ones.

Definition 5 ***(Convergent and Divergent time-points)*** *In a STNU $\mathcal{X} = (V, v_0, E, C)$:*

- *$v_i \in V$ is called a **divergent** time-point iff $\exists j, k, i \neq j \neq k$ with $v_i \rightarrow v_j \in E \cup C$ and $v_i \rightarrow v_k \in E \cup C$;*
- *$v_i \in V$ is called a **convergent** time-point iff $\exists j, k, i \neq j \neq k$ with $v_j \rightarrow v_i \in E \cup C$ and $v_k \rightarrow v_i \in E \cup C$;*
- *V_{dv} is the set of divergent time-points with $V_{dv} \subseteq V$;*
- *V_{cv} is the set of convergent time-points with $V_{cv} \subset V$;*

Intuitively, a divergent node has at least two outgoing edges in the input graph modeling the STNU, and a convergent one has at least two incoming edges.

Please note that if a contingent link is necessarily a directed edge (implicit precedence), a requirement link may be a non directed edge : e.g. $v_i \xrightarrow{[-5,10]} v_j$, imposing some constraint on the temporal distance between the time-points but allowing any order between them at execution time. Hence, v_i or v_j in this example may be considered as a divergent time-point, depending on the order between them in the input link defined at the design level (here the link will be an outgoing edge from v_i). As it will be shown in the next sub-section, it will only change the begin and end points of the two paths that form a cycle, but the cycle will still be the same.

In addition, $V_{dv} \cap V_{cv}$ may not be void, i.e. any $v \in V$ may be convergent, divergent, convergent *and* divergent, or neither convergent nor divergent (we shall call the latter *serial* nodes). In fact, these definitions are orthogonal to the distinction between controllable and contingent time-points, meaning that a controllable time-point might be convergent or divergent, etc., and a contingent one alike (since a contingent and a controllable constraint can converge on the same point, but not two or more contingent : an assumption we make as it's obvious such STNU is not controllable).

Of course, by definition, v_0 cannot be a convergent time-point, but usually, a divergent one, even though the model does not enforce it, as v_0 is used to define the absolute time of any time-point v_i as a constraint between v_0 and v_i.

One can see that such a characterization is very similar to what is done in flow networks [8], but there the problem is to check that the sum of labels (capacities) that converge on a point equals the sum of the labels that exit that node, while here we will instead use this distinction to look for cycles, i.e. identify that two *paths* diverging from one node and reunite in a convergent node have compatible overall durations whatever values the contingent in those paths will take, which is a local WC condition.

In figure 1(a), we present an STNU as defined in definition 1 augmented by definition 5. Figure 1(b) exhibits an alternative way to represent the STNU that will be explained in the next sections.

4.2 Weak controllability on cycles

First, checking WC of $\mathcal{X}$ is straightforward when there is no convergent time-point. Thus $\mathcal{X}$ is a multi-linear graph,

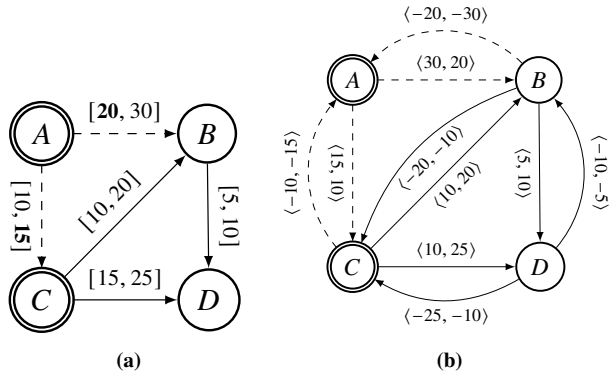


Figure 1 – An STNU is presented in (a) where time-point A plays here the role of the reference point v_0, $V_{dv} = \{A, C\}$ (doubly circled nodes) and $V_{cv} = \{D, B\}$. The links represented by the dotted arrows are contingent constraints. Hence, C and B are uncontrollable time points, while A and D are controllable time points. The proposed STNU is not weakly controllable due to the projection highlighted in bold on the contingent constraints $A \overset{[10,15]}{\dashrightarrow} C$ and $A \overset{[20,30]}{\dashrightarrow} B$ that violates the synchronization on B. We show in (1b the controllable bounds graph of the STNU.

i.e. only divergent simple timelines on which it is always possible to just wait for the observation of an uncontrollable time-point to then decide the activation of the next controllable point, which builds up both the projection and the schedule overtime. In that case, DC and WC are actually equivalent and are always satisfied. When there is at least one convergent point (which entails that there is at least one divergent point), that means there are **paths** that diverge at some point and merge at another point.

Definition 6 ***(Path)*** *A path ρ in $\mathcal{X}$ is a sequence of time-points $v_1, \ldots, v_p$ such that $\forall i = 1 \ldots p-1, v_i \rightarrow v_{i+1} \in E \cup C$ or $v_{i+1} \rightarrow v_i \in E \cup C$, $v_1 \in V_{dv}$ and $v_p \in V_{cv}$.*

One can see that in that definition, we allow a path to follow edges in the graph in any direction, thus ensuring that all possible cycles in the STNU will not be forgotten. For example, in Figure 1(a), considering divergent node C and convergent node D, there is obviously a path C-B-D, but C-A-B-D should also be considered, which is equivalent to stating that there is a path in the corresponding distance graph. Somehow, Figure 1(b), if one disregard, for now, the labels, can be viewed as such a distance graph : the sequence of directed links C-A-B-D is apparent.

Anyway, it is easy to see that any cycle of initial constraints in the input STNU can be defined as a pair of distinct paths as defined above with the same starting $v_1 \in V_{dv}$ and ending $v_p \in V_{cv}$ time-points. It is a peculiar way of defining those cycles that will be useful for our algorithm.

Definition 7 ***(WC Divergent Cycle)*** *A **divergent cycle** $\mathcal{M}$ is a pair (ρ_1, ρ_2) such that ρ_1 and ρ_2 are two **paths** starting at the same divergent time point $v_d \in V_{dv}$ and ending at the same converging time point $v_c \in V_{cv}$, where v_d, v_c are the only common time points in ρ_1, ρ_2, i.e. $\forall v_i \in \rho_1, v_c \neq v_i \neq v_d : v_i \notin \rho_2$.*
A cycle $\mathcal{M}$ is said to be weakly controllable if the sub-STNU restricted to the set of time-points and constraints involved in both paths is WC.

For example, Figure 3b shows a cycle with a first path represented by C-D and a second one by C-B-D.

Then an STNU is WC if and only if all divergent cycles are WC. We will present this result in two steps, first defining a local property that might be checked for a divergent node, and then generalizing to all divergent nodes ; which will mainly be useful for better explaining our algorithm.

Definition 8 ***(Local divergent-WC)*** *Let $\mu(v_d) = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ the set of all cycles starting from $v_d \in V_{dv}$, converging on a set of convergent nodes of V_{cv} that are necessarily ordered (topological ordering) after v_d in the STNU $\mathcal{X}$. We say that $\mathcal{X}$ is **locally divergent-WC on** v_d iff $\forall\, \mathcal{M}_i \in \mu(v_d)$, $\mathcal{M}_i$ is weakly controllable*

For example, Figure 3a shows the cycles starting from the divergent time-point A.

The local divergent-WC does not imply WC, as the corresponding sub-STNU might contain other divergent nodes.

Theorem 1 ***(Global controllability)*** *$\mathcal{X}$ is Weakly controllable (WC) iff $\forall v_d \in V_{dv}$, $\mathcal{X}$ is locally divergent-WC on v_d*

Theorem 1 implies that checking the local divergent-WC property of all the divergent nodes of an STNU is enough to verify the WC.

Proof : The forward implication is straightforward to prove : if there is a divergent node for which at least one divergent cycle, which is a sub-STNU, is not WC, that means there is at least one projection for which there is no consistent local schedule, then the global STNU will not be WC.

For the reverse implication, suppose the global STNU is not WC. Then there is at least one projection for which the corresponding STN is inconsistent ; that is equivalent to having a negative cycle somewhere in that STN ; and that negative cycle necessarily relates time-points that form a divergent cycle in the STNU, which in turn is not WC following Definition 7 [13].

5 Local Weak Controllability

In this section, we show how to check the WC of a cycle by exploiting the convexity of the problem : as proved in [13], considering the lower and upper bounds of the contingents is enough to check WC for STNU.

Definition 9 ***(Controllable Bounds)*** *Given an STNU $\mathcal{X} = (V, v_0, E, C)$, and $t_{ij} \in E \cup C$. The* ***controllable bounds*** *of t_{ij}, denoted Π_{ij}^{ctl}, is the tuple of discrete values such that :*

$$\Pi_{ij}^{ctl} = \langle min_{ij}^{ctl}, max_{ij}^{ctl} \rangle$$

where, min_{ij}^{ctl} and max_{ij}^{ctl} respectively represent the minimal and maximal duration that can be guaranteed for t_{ij}.

It is easy to see that any requirement constraint $e_i = [l_{ij}, u_{ij}]$, has a minimal and maximal duration that can be guaranteed with $min_{ij}^{ctl} = l_{ij}$ and $max_{ij}^{ctl} = u_{ij}$.

A contingent constraint is different due to its uncontrollable duration. Still, some guarantee can be extracted on its execution. Indeed, the duration of any $c_i \in C$ is guaranteed, for the minimum, to be at most equal to U_{ij} and, for the maximum, at least equal to l_{ij}. Thus, we have : $min_{ij}^{ctl} = u_{ij}$ and $max_{ij}^{ctl} = l_{ij}$. Intuitively, min_{ij}^{ctl} and max_{ij}^{ctl} represent the two worst-case scenarios of a contingent duration. In the following, we generalize Π_{ij}^{ctl} as follows :

$$\Pi_{ij}^{ctl} = \begin{cases} \langle u_{ij}, l_{ij} \rangle \text{ iff } t_{ij} \in C \\ \langle l_{ij}, u_{ij} \rangle \text{ iff } t_{ij} \in E \end{cases} \quad (1)$$

Then, from Equation 1, it is actually possible to represent an STNU $\mathcal{X}$ in terms of its controllable bounds graph denoted $\Pi_{\mathcal{X}}^{ctl}$, which is shown in Figure 1 (b). This graph considers each original constraint and its inverse. A requirement constraint $e_i = [l_{ij}, u_{ij}]$, equivalently $(v_j - v_i) \geq l_{ij}$ and $(v_j - v_i) \leq u_{ij}$, has an inverse constraint $e_i' : (v_i - v_j) \leq -l_{ij}$ and $(v_i - v_j) \geq -u_{ij}$ equivalently represented as $e_i' = [-u_{ij}, -l_{ij}]$. The same transformation is applied to contingent constraints. Please note that we introduce this new graph representation instead of reusing the well-known *distance graph* of an STNU as it's simpler. We also believe this new representation could pave the way for more efficient algorithms for DC.

From this transformation, it's possible to compute the controllable bounds of a path ρ composed of constraints in $E \cup C$ by controllable bounds propagation from v_1 to v_p.

Definition 10 ***(Controllable Path Bounds)*** *Let ρ be a path in $\Pi_{\mathcal{X}}^{ctl}$, with $v_1, \ldots, v_p$ the sequence of time-points of ρ. The* ***controllable path bounds*** *denoted Π_{ρ}^{ctl} is defined as follows :*

$$\Pi_{\rho}^{ctl} = \langle \sum min_{ij}^{ctl}, \sum max_{ij}^{ctl} \rangle$$

From this point, it's possible to check the WC controllability of a cycle $\mathcal{M} = (\rho_1, \rho_2)$ through the controllable paths bounds $\Pi_{\rho_1}^{ctl}$ and $\Pi_{\rho_2}^{ctl}$. Indeed, we need to guarantee that the minimum controllable duration of ρ_1 is lesser or equal to the maximum controllable duration of ρ_2 and vice-versa. Intuitively, if the condition is not satisfied, then there exists a projection of $\mathcal{M}$ such that ρ_1 and ρ_2 cannot synchronize on v_p as Π_{ρ}^{ct} represent the worst-case scenarios of ρ. the worst-case scenarios for synchronizing two paths are those where, for one path, its contingents take the minimal bound l_{ij} and the maximal bounds u_{ij} for the second path.

Theorem 2 ***(Cycle WC property)***

Given a cycle $\mathcal{M} = (\rho_1, \rho_2)$ and the controllable paths bounds $\Pi_{\rho_1}^{ctl} = \langle min_{\rho_1}^{ctl}, max_{\rho_1}^{ctl} \rangle$ and $\Pi_{\rho_2}^{ctl} = \langle min_{\rho_2}^{ctl}, max_{\rho_2}^{ctl} \rangle$. $\mathcal{M}$ is weakly controllable iff :

$$(min_{\rho_1}^{ctl} \leq max_{\rho_2}^{ctl}) \wedge (min_{\rho_2}^{ctl} \leq max_{\rho_1}^{ctl}) \quad (2)$$

Proof : The forward implication is straightforward to prove : if $\mathcal{M}$ is WC, then whatever the bounds of the contingents in $\mathcal{M}$, there always exists a schedule that satisfies the constraints of $\mathcal{M}$. Let's suppose Equation 2 is false. It means there exists a projection of ρ_1 and ρ_2 such that the synchronization on v_p is impossible and forms a negative cycle. Thus, such a projection is inconsistent, and $\mathcal{M}$ is not WC. This is impossible as all projections of $\mathcal{M}$ are consistent.

For the reverse implication, let's suppose $\mathcal{M}$ is not WC, but Equation 2 is satisfied. Then, it means that the projections of the two worst-case scenarios of $\mathcal{M}$ are consistent as there exists at least one schedule that guarantees the synchronization on v_p. Thus, any projection satisfies the synchronization on v_p. This is not possible as $\mathcal{M}$ is not WC, which implies the sub-STNU M has a negative cycle [13].

For simplicity, we denote M^{ctl} a worst-case scenario of $\mathcal{M}$. A running example would be considering the network on the left of Figure **??** as an STNU. The controllable bounds are : $\{30, 20\}$ for $A \xrightarrow{[20,30]} B$, $\{15, 10\}$ for $A \xrightarrow{[10,15]} C$, and $\{10, 20\}$ for $C \xrightarrow{[10,20]} B$. Only one cycle exists that is formed by the two paths (ρ_1, ρ_2) from A to B, where $\Pi_{\rho_1}^{ctl} = \{30, 20\}$ and $\Pi_{\rho_2}^{ctl} = \{25, 30\}$, which does not satisfy Equation 2 as $min_{\rho_2}^{ctl} > max_{\rho_1}^{ctl}$.

6 The WC-Checking algorithm

6.1 Description of the algorithm

In this section, we present the new WC-checking algorithm for STNU by finding and checking its cycles. The algorithm comprises two parts : the first one finds the cycles of a divergent time-point, and the second one checks those cycles. In the following, we formalize the basic structures for the WC checking algorithm given an STNU $\mathcal{X}$:

— a **projection path** p, is a path $\langle \alpha_p, C_p, V_p \rangle$ in $\Pi_{\mathcal{X}}^{ctl}$ from v_d to v_m, where α is the minimal or maximal controllable bound of Π_{p}^{ctl} obtained by propagation. C_ρ is the set of contingent constraints of p ($C_p \subseteq C$),

and V_p the set of time-points of p ($V_p \subseteq V$). Please note that p is the definition of a path from an algorithm point of view, while in reality, p and ρ are equivalent.

— $P(v_d)$ is a set of **projection paths** $P(v_d) = \{p_i, \ldots, p_m\}$ that start at a divergent time-point v_d and still have to converge on a common v_c.

— the **minimal divergent cycles** $D_{min}(v_d)$ is a mapping of convergent time points v_c to a set of projection paths ($P^{min}_{v_c}$) that converge from v_d to v_c such that $\forall p \in P_{v_c}, \alpha_p = min^{ctl}_p$.

— the **maximal divergent cycles** $D_{max}(v_d)$ is a mapping of convergent time points v_c to a set of projection paths ($P^{max}_{v_c}$) that converge to v_c from v_d such that $\forall p \in P_{v_c}, \alpha_p = max^{ctl}_p$.

We introduce in Algorithm 1 the *findDivergentCycles* algorithm in charge of finding the cycles of a divergent time-point v_d. To avoid going through all possible paths in the controllable bounds graph $\Pi^{ctl}_{\mathcal{X}}$, we prune the number of paths in two ways :

— We first add the notion of *rank*, which was not formally defined in our model as it was not needed but is common in qualitative temporal networks : it is possible to define a partial order of all time-points with regard to the precedence relation ; $rank(v_0) = 0$, then for all v_i such that $v_0 \preceq v_i \in E \cup C$ and there is no v_j such that $v_0 \preceq v_j \in E \cup C$ and $v_j \preceq v_i \in E \cup C$, $rank(v_i) = 1$, and so on and so forth.

— Using that rank, a forward search is then applied by ordering the time-points through a topological ordering algorithm from v_0 (rank 0). This enables us to avoid any time-point v_i with a lower rank than the current divergent time-point v_d.

— A distinction between the minimal and maximal controllable bounds of a path. We apply two forward searches : one that computes the paths with only the maximal controllable bound and one with the minimal controllable bound. This allows us to prune the paths that converge to any convergent time-point to keep only stricter ones. For example, it is easy to see that for two paths p_i and p_j such that $C_{p_i} = C_{p_j} = \{\emptyset\}$ (only requirement constraints) p_i is stricter than p_j if $min^{ctl}_{p_i} > min^{ctl}_{p_j}$ (respectively, $max^{ctl}_{p_i} < max^{ctl}_{p_j}$). Hence, it's useless to consider further p_j as p_i is a stricter path, and only p_i is kept in $D_{min}(v_d)$ or $D_{max}(v_d)$ depending on the computed controllable bound. This also holds for a path p_j such that $C_{p_j} \neq \{\emptyset\}$ (with contingent constraints). However, when C_{p_i} and C_{p_j} are not empty, it's impossible to apply these rules as it might result in removing an inconsistent cycle in the graph. Suppose we have the minimal controllable bounds of p_i and p_j and the maximal controllable bounds of a path p_k such that the tuple (p_j, p_k) is the only one to form a cycle M^{ctl}. Then, if ρ_i is stricter than p_j and p_j is not kept, M^{ctl} will never be checked likewise for the WC of $\mathcal{X}$. Therefore, both p_i and p_j must be kept in $D_{min}(v_d)$. This is actually the reason why full reduction of intervals through the intersection of different edges is not possible, and hence, a polynomial time algorithm cannot be found, unlike DC and SC.

From lines 1 to 3, we initialize the maps $D_{max}(v_d)$ and $D_{min}(v_d)$, and the set of paths P. Then, the core of the algorithm (lines 5-16) propagates the paths in P to find and keep all stricter paths of v_d in $D_{max}(v_d)$ until P = $\{\emptyset\}$. In fact, in line 14, we also update P(v_d) and P_{v_j} by removing the paths that are not stricter anymore. A second forward search is done for $D_{min}(v_d)$ where P(v_d) is reset. Once the forward searches are over, the maps $D_{max}(v_d)$ and $D_{min}(v_d)$ contain all the restrictive paths from v_d to a convergent time-point v_c. Then, we execute the *checkCycles* algorithm (see Algorithm 2) in charge of checking the WC of the cycles of v_d. This algorithm is trivial as it simply searches and checks for each v_c in $D_{max}(v_d)$ and $D_{min}(v_d)$ all the pairs of paths (p_{min}, p_{max}) that converge on v_c and form a cycle $\mathcal{M}^{ctl}$ where $V_{p_{min}} \cap V_{p_{max}} = \{v_d, v_c\}$.

Algorithm 1: findDivergentCycles algorithm

Input: v_d :(time-point), $\Pi^{ctl}_{\mathcal{X}}$ **: (graph), rank : map**
Output: Boolean

```
1  D_min(v_d) = { }
2  D_max(v_d) = { }
3  P(v_d) = [⟨0, [], [v_d]⟩]
4  A first forward search for D_max
5  while P not empty do
6      p = P(v_d)[0]   p is removed in P(v_d)
7      for each child v_j of v_m ∈ V_p with rank(v_j) ≥
         rank(v_d) and v_j ∉ V_p do
8          p = propagateMaxPath(Π^ctl_X, p, max^ctl_mj)
9          if v_j is a convergent time point (v_j ∈ V_c)
             then
10             if v_j not in D_max(v_d) then
11                 add v_j → [p] in D_max(v_d)
12             else
13                 if p is a restrictive path in P_{v_j} then
14                     add p to P_{v_j} and to P(v_d)
15         else
16             add p to P
17 A second forward search for D_min(v_d)
18 return checkCycles(D_max(v_d), D_min(v_d))
```

Finally, Algorithm 3 presents the *WC-checking* algorithm that, for a given STNU $\mathcal{X}$, computes its controllable bounds graph $\Pi^{ctl}_{\mathcal{X}}$ (line 1), determines the topological ordering of

Algorithm 2: checkCycles algorithm

Input: $D_{max}(v_d)$, $D_{min}(v_d)$
Output: Boolean

```
1 for each v_c → P^min_{v_c} in D_min(v_d) do
2     for each p_min in P^min_{v_c} do
3         for each p_max in P^max_{v_c} in D_max(v_d) do
4             if (p_min, p_max) is of the form M^ctl then
5                 if α_{p_min} > α_{p_max} then
6                     return False   Or the cycle
7 return True
```

the time-points (line 2), and find and check the cycles of each divergent time-point in V_d. We inform the readers that the algorithm does not need to check the pseudo-controllability of Morris [11] as long as the constraint bounds l_{ij}, u_{ij} are as follows : $l_{ij} \neq -\infty$ and $u_{ij} \neq +\infty$.

We show, in a simplified manner, the execution of our WC-Checking algorithm in Figure 3. The order from the ranking is $A \rightarrow C \rightarrow B \rightarrow D$ and is used in the forward searches for finding and checking the cycles from A and C. We show in Figure 2 a running example of our algorithm with the divergent time-point A. Please note that for the sake of this paper, we simplified the example.

Algorithm 3: WC-Checking algorithm

Input: $\mathcal{X}$: STNU(V,v_0, V_c,V_d,E,C)
Output: Boolean

```
1 Π^ctl_X = getDistanceGraph(X)
2 rank = orderFromRank(X)
3 for each v_d in V_d do
4     if findDivergentCycles(v_d, Π^ctl_X, rank ) ==
       False then
5         return False  Or non-WC cycles of v_d
6 return True   Or all non-WC cycles
```

6.2 Features and Complexity

The previous section presented the new WC-checking algorithm for STNU that checks WC by checking the internal cycles of an STNU. The particularity of this approach easily allows the algorithm to return the set of negative cycles of a non-weakly controllable STNU (see Algorithms 2, 3). This feature is important for explainability, as explained in Section 1. Moreover, the divergent time-points are independent, which makes parallelization over the divergent time-points possible. In addition, the pseudo-controllability step is not required for constraint bounds with finite values (no constraints with a $+\infty$ or $-\infty$ for bounds). Thus, the algorithm can be executed incrementally by checking only divergent time points of the same rank or lesser (topological ordering) than the starting time point of the newly added constraint is enough due to divergent time-points being independent. However, it's not optimal as it might check cycles that are not linked to the newly added constraint. The drawback of the algorithm is that the minimal network of weakly controllable STNUs is not computed.

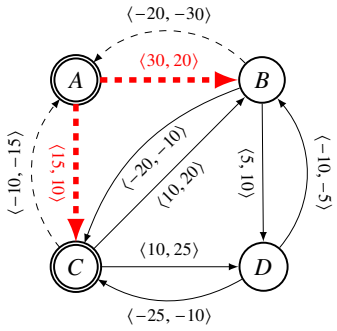
The temporal complexity of the algorithm depends on the number of cycles to check, which is related to multiple parameters such as the number of contingents, the number of divergent time-points, and the number of successors per divergent time-point. For a complete graph, the algorithm is exponential, and its complexity is not better than the original algorithm ($2^{|C|}$). However, our interest lies in realistic graphs where the density of the graph is low. Thus, in the next section, we will compare our algorithm (new_WC) with the original WC-checking algorithm (old_WC) (see Section 2) by restricting the parameters. We will also compare it with the Floyd-Warshall algorithm (APSP) to compare the behavior of our algorithm to see if a polynomial behavior is possible when parameters are restricted enough. Please note that the APSP cannot check the WC of STNU.

7 Experiments

In order to empirically test the effectiveness of the proposed algorithm, we consider the time from the start of the algorithm execution until it has finished all computation (not simply after finding an STNU to be weakly controllable or not). The benchmark comes from a random generator we implemented that can randomly generate sparse STNUs. It creates an STNU in the form of a complete directed acyclic graph (DAG), from which we randomly and remove a number of edges depending on parameters : the number of time points n, the rate of divergent time points r_d and the number of successors n_c, and the rate of contingency r_c.

All the experiments have been performed on a machine equipped with an Intel Core processor : 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz 2.50 GHz. We used a time/memory limit of 10 minutes/4GB and sequential, single-core computation for the sake of the experiments.

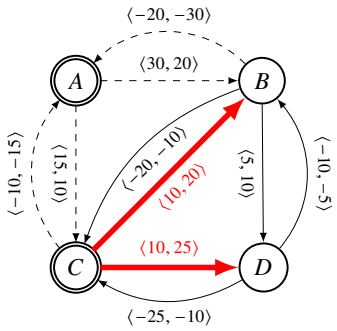
We experiment under different settings : n = $\{20, 50, 100, 200, 500, 1000\}$, $r_d = \{0.1, 0.2, 0.3\}$ meaning 10 to 30% of divergent time-points, $r_c = \{0.2, 0.3\}$, and $n_c = 3$. For each combination of parameters, we generate 20 STNUs and compute the average execution time. We show in Figure 4a that, in general, our algorithm clearly outperforms the *old-WC* algorithm and is slightly worse than the APSP algorithm up to 20% of contingent constraints. This shows that the parameters were bounded enough to have a polynomial behavior. However, beyond this threshold, our algorithm starts to show its limit. This shows the sensitivity of our algorithm to the parameters (see Figure 4b).



Step 1

$D_{min} = \{C : \langle 15, AC\rangle, \text{B} : \langle 30, AB\rangle\}$

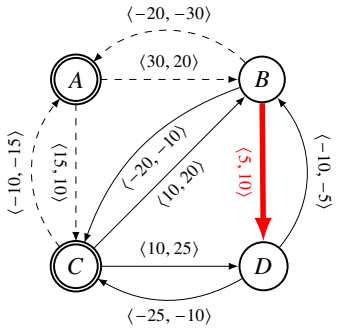
$D_{max} = \{C : \langle 10, AC\rangle, \text{B} : \langle 20, AB\rangle\}$



Step 2

$D_{min} = \{C : \langle 15, AC\rangle, \text{B} : [\langle 20, AB\rangle, \langle 25, ACB\rangle], \text{D} : \langle 15, ACD\rangle\}$

$D_{max} = \{C : \langle 10, AC\rangle, \text{B} : [\langle 20, AB\rangle, \langle 30, ACB\rangle], \text{D} : \langle 25, ACD\rangle\}$



Step 3

$D_{min} = \{C : \langle 15, AC\rangle, B : [\langle 20, AB\rangle, \langle \mathbf{25}, \mathbf{ACB}\rangle],$
$D : [\langle 15, ACD\rangle, \langle 35, ABD\rangle, \langle 30, ACBD\rangle]\}$

$D_{max} = \{C : \langle 10, AC\rangle, B : [\langle \mathbf{20}, \mathbf{AB}\rangle, \langle 30, ACB\rangle],$
$D : [\langle 25, ACD\rangle, \langle 30, ABD\rangle, \langle 40, ACBD\rangle]\}$

Figure 2 – This Figure shows, in a simplified manner, a running example of Algorithm 1 with divergent time-point A by showing only the value and the time-points of a path p. In addition, we highlight the edges taken at each step in the forward search (that respect the conditions in line 7 of Algorithm 1). After step 3, D_{min} and D_{max} contain all the restrictive paths (only those that need to be kept). Then, with Algorithm 2, we find the couples of paths that form a cycle and check if they are WC. It is easy to see that the paths that form the cycles for A (see Figure 3a) are present in D_{min} and D_{max} (we highlight the one that is not WC).

8 Conclusion

This paper introduced a novel approach for checking the WC of an STNU by checking the consistency of its elementary cycles. Interesting features of our algorithm to consider further are as follows : it can identify the constraints causing the uncontrollability, and it can be executed in an incremental way (not optimal) and in a parallelized way. However, it is not capable of computing the minimal network of an STNU. Moreover, we exhibited that the algorithm's complexity depends on the sparsity of the STNU, which makes it exponential in the worst cases. However, experiments show that in loosely connected STNU, the algorithm tends to behave in a polynomial-like way. Finally, the paper argues the relevance of the problem of WC in a multi-agent setting, where uncontrollable events are not controlled by nature but by other agents in the system. Further work will tackle the problem of repairing negative cycles by negotiating the duration of the uncontrollable events, whose duration depends on the other agents' decisions.

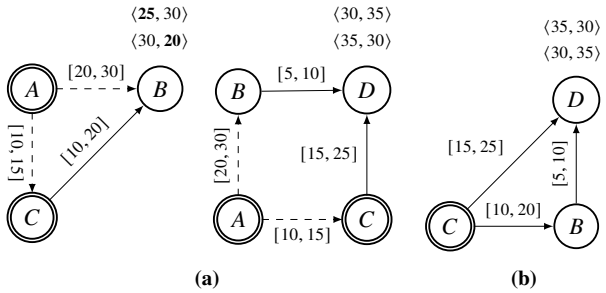
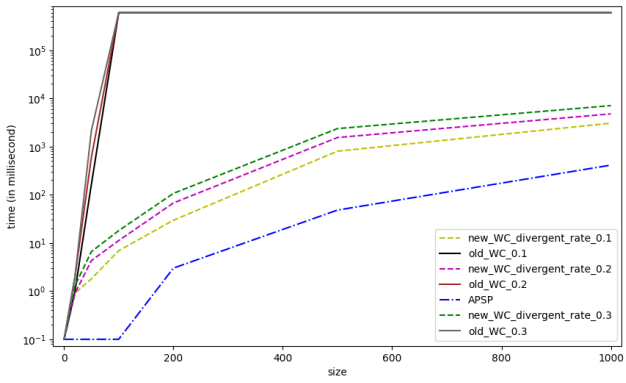
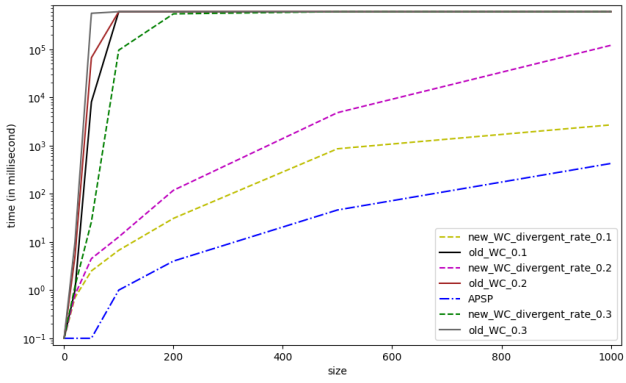


FIGURE 3 – The figure represents, in a simplified way, the steps of the algorithm for the example of Figure 1. We show in Figure 3a the cycles of the divergent time-point A that are found and checked by the algorithm. The same is done for the divergent time-point C in Figure 3b. The STNU in Figure 1 is not weakly controllable as the cycle ABC is not weakly controllable due to equation (15 + 10) ≤ 20, highlighted in bold, being false.



(a)



(b)

FIGURE 4 – Experimentation with 20% (a), and 30% (b) of contingent constraints

Références

[1] Shyan Akmal, Savana Ammons, Hemeng Li, and James C Boerkoel Jr. Quantifying degrees of controllability in temporal networks with uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2019.

[2] Shyan Akmal, Savana Ammons, Hemeng Li, Michael Gao, Lindsay Popowski, and James C. Boerkoel. Quantifying controllability in temporal networks with uncertainty. *Artificial Intelligence*, 2020.

[3] Arthur Bit-Monnot and Paul Morris. Dynamic controllability of temporal plans in uncertain and partially observable environments. *J. Artif. Intell. Res.*, 2023.

[4] Guillaume Casanova, Cédric Pralet, Charles Lesire, and Thierry Vidal. Solving dynamic controllability problem of multi-agent plans with uncertainty using mixed integer linear programming. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, 2016.

[5] Alessandro Cimatti, Andrea Micheli, and Marco Roveri. Solving temporal problems using smt : weak controllability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2012.

[6] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 1991.

[7] Luke Hunsberger and Roberto Posenato. Speeding up the rul dynamic-controllability-checking algorithm for simple temporal networks with uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[8] Jsen-Shung Lin, Chin-Chia Jane, and John Yuan. On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. *Networks*, 1995.

[9] Josef Lubas, Marco Franceschetti, and Johann Eder. Resolving conflicts in process models with temporal constraints. In *Proceedings of the ER Forum and PhD Symposium*, 2022.

[10] Paul Morris. Dynamic controllability and dispatchability relationships. In *Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*. Springer, 2014.

[11] Paul H. Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 2005.

[12] Ajdin Sumic, Andrea Micheli, Alessandro Cimatti, and Thierry Vidal. Smt-based repair of disjunctive temporal networks with uncertainty : Strong and weak controllability. In *Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31 , 2024. Proceedings*, 2024.

[13] Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks : from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.*, 11(1) :23–45, 1999.