A More Efficient and Informed Algorithm to Check Weak Controllability of Simple Temporal Networks with Uncertainty

Ajdin Sumic
Technological University of Tarbes, France

Thierry Vidal
Technological University of Tarbes, France

Abstract

Simple Temporal Networks with Uncertainty (STNU) are a well-known constraint-based model expressing sets of activities (e.g., a schedule or a plan) 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: the Strong and Dynamic Controllability (SC/DC) has proven both useful in practice and provable in polynomial time, while Weak Controllability (WC) is co-NP-complete and has been left aside. Moreover, controllability checking algorithms are propagation strategies, which have the usual drawback, in case of failure, to prove unable to locate the contingents that explain the source of 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; (2) it provides a new WC-checking algorithm whose performance in practice depends on the network structure and is faster in loosely connected ones; (3) it provides the failing cycles in the network that explain non-WC.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Temporal constraints satisfaction, uncertainty, STNU, Controllability checking, Explainable inconsistency, Multi-agent planning

Digital Object Identifier 10.4230/LIPIcs.TIME.2024.8

1 Introduction and Related Work

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)[5]: 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.

An STN with Uncertainty (*STNU*) is an extension in which one distinguishes a subset of constraints whose effective duration is not assigned but observed (uncontrollable durations). This is useful for addressing realistic dynamic and stochastic domains where such durations are usually set by the environment.

© Ajdin Sumic and Thierry Vidal;
licensed under Creative Commons License CC-BY 4.0

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 are taken by the contingent durations. In [14], the authors introduce three levels of controllability that express how and when the uncertainties are resolved: the Weak Controllability (WC) proves that 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; Dynamic Controllability (DC) assumes that at execution time, a strategy can be built based on past observations only thus, whatever the contingent durations still to be observed; Strong Controllability (SC) is 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 unrealistic in dynamic applications that assume full progressive observability at execution time, where DC looks more relevant and have received much attention in previous works. In contrast, SC fits perfectly application domains with 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 specifically designed propagation-based algorithms that run in polynomial time [11, 3, 14]. While WC is a co-NP-complete problem [12], and only exponential algorithms exist to check WC [4, 14]. This is another reason why WC has been disregarded [2, 14].

This paper tackles Weak Controllability by first exhibiting its relevance in several contexts (e.g., multi-agent task management) and providing a more efficient algorithm for realistic networks, i.e., loosely connected networks. Contrary to the complete propagation algorithms proposed for SC and DC, our algorithm maintains and reasons only on the input constraints, which form network paths. 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. This way, the algorithm can also diagnose the source of uncontrollability of a non-WC STNU by detecting the set of constraints (here, cycles) that make the STNU not Weakly controllable. This explainability issue was recently addressed and is important to repair non-controllable STNUs [9, 2, 1, 13].

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

A Simple Temporal Network (STN) 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 [5], in the form $v_j - v_i \in [l_{ij}, u_{ij}]$, with lower bounds $l_{ij} \in \mathbb{R} \cup \{-\infty\}$ and upper bounds $u_{ij} \in \mathbb{R} \cup \{+\infty\}$. Interestingly enough, this model encompasses the qualitative precedence constraint, since v_i *precedes* v_j, noted $v_i \preceq v_j$, iff $l_{ij} \geq 0$. 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, i.e., to assign 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 [14].

▶ **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) and where v_0 is 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 relates two time-points $e_k = v_j - v_i \in [l_{ij}, u_{ij}]$ with, $v_i, v_j \in V$.*
- *C a set of contingent constraints $\{c_1, \ldots, c_{|C|}\}$, where each c_k relates two time-points $c_k = v_j - v_i \in [l_{ij}, u_{ij}]$ 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 (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, having a contingent duration between two unordered time-points is semantically impossible. Figure 1a is the graphical representation of an STNU.

In addition, an STN (and hence an STNU too) has an equivalent *distance graph* representation [5, 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 $v_j \xrightarrow{-l} v_i$.

In STNUs, consistency has been redefined through three levels of *controllability*, which we will recall hereafter before focusing on one of them, namely the Weak controllability.

▶ **Definition 2** (Schedule). *A schedule δ of an STNU $\mathcal{X}$ is the assignment of one value for each controllable time-point $\delta = \{\delta(v) \mid v \in V_c\}$.*

▶ **Definition 3** (Situation and Projection). *Given an STNU $\mathcal{X}$, the **situations** of $\mathcal{X}$ is a set of tuples Ω defined as the cartesian product of contingent domains:*

$$\Omega = \bigtimes_{c \in C} [l_c, u_c]$$

*A **situation** is an element ω of Ω and we write $\omega(c)$ with $c \in C$ to indicate the element in ω associated with c in the cross product. A **projection** $\mathcal{X}_\omega = (V, E \cup C_\omega)$ of $\mathcal{X}$ is an STN where $C_\omega = \{[\omega(c), \omega(c)] \mid c \in C\}$. Last, a schedule δ_ω which satisfies all the constraints in $\mathcal{X}_\omega$ is called a **solution** of $\mathcal{X}_\omega$.*

Intuitively, the set of situations defines the space of uncertainty, i.e., the possible values of contingent constraints; a projection substitutes all contingent links with a singleton, forcing its duration to the value appearing in ω. Now, a network shall be deemed controllable if it is possible to schedule the controllable time points to satisfy all requirement constraints in any possible projection. But that depends on how and when the contingent durations are observed/known by the execution supervisor.

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

This definition implies that an "oracle" communicates contingents' durations to the scheduler before execution time, which requires all projections to be independently consistent.

We provide the two other controllability levels only for the sake of completeness, though they will not be addressed in this paper. Dynamic controllability (DC) demands that the assignment of a controllable time-point only depends on past observations, and Strong controllability (SC) demands a unique schedule that is totally independent from any observation [14].

▶ **Definition 5** (Dynamic Controllability (DC)). *An STNU $\mathcal{X}$ is* ***Dynamically controllable*** *iff it is Weakly controllable and* $\forall v_i \in V_c, \forall \omega, \omega' \in \Omega$, $\omega^{\preceq v_i} = \omega'^{\preceq v_i} \implies \delta_\omega(v_i) = \delta'_\omega(v_i)$ *where* $\omega^{\preceq v} = \{\omega_k \in \omega \text{ s.t. } end(c_k) \preceq v\}$ *is the part of the situation* ω *which contingent constraints ending time-points precede* v*.*

▶ **Definition 6** (Strong Controllability (SC)). *An STNU $\mathcal{X}$ is* ***Strongly controllable*** *iff* $\exists\, \delta$ *such that* $\forall \omega \in \Omega, \delta$ *is a solution of* $\mathcal{X}_\omega$*.*

As said before, polynomial-time propagation-based checking algorithms exist for SC and DC [14][11][3]. But not for WC checking, which is co-NP-complete [12]. 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. This is enough to check WC as it has been proven in [14] that considering only the bounds of contingents is enough to verify any level of controllability in STNUs.

3 Relevance of Weak Controllability

In this section, we will argue that WC may be more relevant than DC and SC for some applications and, thus, deserves to be investigated.

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. For instance, the duration of a truck ride to deliver some goods depends on exogenous traffic conditions that no one has control over. There, the real duration will be observed only at execution time, which calls for DC enforcement. However, in many domains (logistics, transport, services), one may 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 construction site, one needs a weekly plan for visiting patient rooms or for construction tasks. Still, the assigned teams (number of people, skills) are unknown, resulting in flexible and large enough intervals of possible durations. The precise assignment is only known each day for the next day, which allows for 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 acticity duration might be controlled by another agent instead of Nature. Thus, some tasks might be controllable (requirement) for one agent but uncontrollable for another (contingent). For instance, in collaborating hospital services that share common resources: one service might need to wait before another one sends a patient. For the other agent controlling the duration, that represents a degree of freedom, i.e., the *flexibility*, that some agent wishes to keep as long as possible to be more robust. Then, collaboration may rely on the timely communication of effective durations at execution time. But it is also possible that they plan in advance their weekly operations with maximum flexibility but must set their own schedules each day for the next one. They will communicate their decisions to the agents that depend on them, for better coordination. Therefore checking WC instead of DC/SC enables the agents to be more robust through least-commitment strategies.

4 From local controllability to global controlability

4.1 Updated STNU graphical model

A starting point for resolving the issue of WC 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 but also by *divergent* time-points and *convergent* ones. From Definition 7, 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.

▶ **Definition 7** (Convergent and Divergent time-points). *In a STNU* $\mathcal{X} = (V, E, C)$ *:*

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

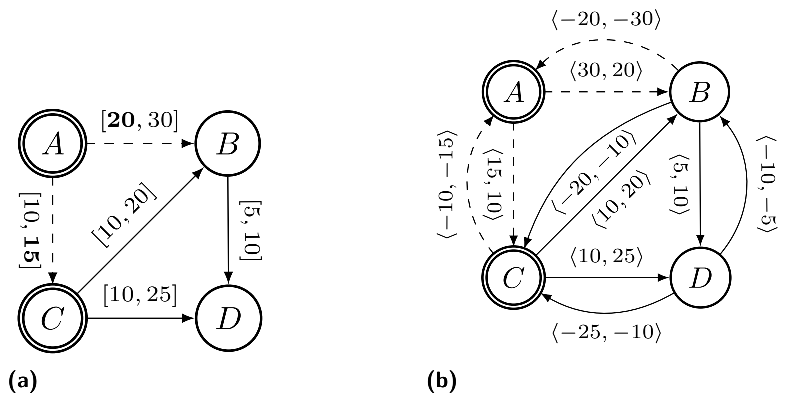


Figure 1 An STNU is presented in (a) where time-point A can be seen as the reference point v_0, $V_{dv} = \{A, C\}$ (doubly circled nodes) and $V_{cv} = \{D, B\}$. Dotted arrows express contingent constraints. Hence, C and B are uncontrollable time points, while A and D are controllable ones. The STNU 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 violate the synchronization on B. We show in 1b the controllable bounds graph of the STNU.

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, in this example, v_i or v_j may be considered 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 shown in the next subsection, the beginning and end points of the two paths that form a cycle will only change, but the cycle will still remain.

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 : these definitions are orthogonal to the distinction between controllable and contingent time-points, i.e., a controllable time-point might be convergent or divergent, etc., and an uncontrollable one alike.

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]. Still, 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. Here, we will instead use this distinction to look for cycles, i.e., identify that two *paths* which diverge from one node and reunite in a convergent node have compatible overall durations whatever values the contingents 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 7. Figure 1(b) exhibits an alternative way to represent the STNU that will be explained later.

4.2 Weak controllability on cycles

First, we assume there is at least one convergent point (and hence at least one divergent point). Otherwise the STNU is necessarily WC since there is no cycle among the input constraints and hence no negative one. That means there are **paths** that diverge at some point and merge at another point.

▶ **Definition 8** (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 $v_p \in V_{cv}$.*

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 disregards, for now, the labels, can be viewed as such a distance graph, where the path C-A-B-D appears.

Then, any cycle of input constraints in the STNU can be defined as a pair of distinct paths 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 9** (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. $\rho_1 \cap \rho_2 = \{v_d, v_c\}$.*
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, in Figure 1a one has a cycle (ρ_1, ρ_2) with $\rho_1 =$ A-B and $\rho_2 =$ A-C-B.

Then, an STNU is WC 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 be useful for better explaining our algorithm.

▶ **Definition 10** (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 3d shows the two cycles starting from the divergent time-point A.

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

▶ **Theorem 11** (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 11 implies that checking the local divergent-WC property of all the divergent nodes of an STNU is enough to check the global WC.

Proof. The forward implication is straightforward to prove: if there is a divergent node for which at least one divergent cycle (sub-STNU) is not WC, that means there is at least one projection for which there is no consistent local schedule. Hence, the 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[14]; and that negative cycle necessarily relates time-points that form a divergent cycle in the STNU, which in turn is not WC following Definition 9. ◀

5 Local Weak Controllability

In this section, we show how to check the local WC of a cycle by exploiting the convexity of the problem, only considering the contingents bounds [14].

▶ **Definition 12** (Controllable Bounds). *Given an STNU $\mathcal{X} = (V, E, C)$, and $v_j - v_i \in E \cup C$. The* ***controllable bounds*** *of $v_j - v_i$, denoted Π^{ctl}_{ij}, is the pair of discrete values*

$$\Pi^{ctl}_{ij} = \langle min^{ctl}_{ij}, max^{ctl}_{ij} \rangle$$

where, min^{ctl}_{ij} and max^{ctl}_{ij} respectively represent the minimal and maximal duration that can be guaranteed for $v_j - v_i$.

Any requirement constraint $e_k = [l_{ij}, u_{ij}]$ has a minimal and maximal duration that can be guaranteed with $min^{ctl}_{ij} = l_{ij}$ and $max^{ctl}_{ij} = u_{ij}$. For a contingent constraint $c_k \in C$, we cannot guarantee that at execution time its duration will be lower (resp. greater) than its maximum bound u_{ij} (resp. its minimal bound l_{ij}). Hence, we have $min^{ctl}_{ij} = u_{ij}$ and $max^{ctl}_{ij} = l_{ij}$. Intuitively, e.g., min^{ctl}_{ij} is the worst-case scenario for a contingent duration when trying to control the maximum possible total duration of a path it belongs to. We generalize Π^{ctl}_{ij} as follows:

$$\Pi^{ctl}_{ij} = \begin{cases} \langle u_{ij}, l_{ij} \rangle \text{ iff } v_j - v_i \in C \\ \langle l_{ij}, u_{ij} \rangle \text{ iff } v_j - v_i \in E \end{cases} \tag{1}$$

Then, from Equation 1, it is actually possible to represent an STNU $\mathcal{X}$ in terms of its ***controllable bounds graph*** denoted $\Pi^{ctl}_{\mathcal{X}}$, similar to a distance graph but more suited to our algorithm, which is shown in Figure 1 (b). This graph considers each original constraint and its inverse. A requirement constraint $e_k = [l_{ij}, u_{ij}]$, equivalently $l_{ij} \leq (v_j - v_i) \leq u_{ij}$, has an inverse constraint e'_k: $-u_{ij} \leq (v_i - v_j) \leq -l_{ij}$ equivalently represented as $e'_i = [-u_{ij}, -l_{ij}]$. The same transformation is applied to contingent constraints.

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

▶ **Definition 13** (Controllable Path Bounds). *Let ρ be a path in $\Pi^{ctl}_{\mathcal{X}}$, with $v_1, \ldots, v_p$ the sequence of time-points of ρ. The* ***controllable path bounds*** *denoted Π^{ctl}_{ρ} is defined as follows:*

$$\Pi^{ctl}_{\rho} = \langle \sum min^{ctl}_{ij}, \sum max^{ctl}_{ij} \rangle$$

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 $\Pi^{ctl}_{\rho_1}$ and $\Pi^{ctl}_{\rho_2}$. Indeed, we need to guarantee that the minimum controllable duration of ρ_1 is less than 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^{ct}_{ρ} represent the worst-case scenarios of ρ: the worst cases for synchronizing two paths are when, for one path, its contingents take their minimal bounds l_{ij} and for the second one, their maximal bounds u_{ij}.

▶ **Theorem 14** (Cycle WC property). *Given a cycle $\mathcal{M} = (\rho_1, \rho_2)$ and the controllable paths bounds $\Pi^{ctl}_{\rho_1} = \langle min^{ctl}_{\rho_1}, max^{ctl}_{\rho_1}\rangle$ and $\Pi^{ctl}_{\rho_2} = \langle min^{ctl}_{\rho_2}, max^{ctl}_{\rho_2}\rangle$, $\mathcal{M}$ is weakly controllable iff:*

$$(min^{ctl}_{\rho_1} \leq max^{ctl}_{\rho_2}) \wedge (min^{ctl}_{\rho_2} \leq max^{ctl}_{\rho_1}) \tag{2}$$

Proof. 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.

For the reverse implication, let us 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 has a negative cycle [14]. ◀

Obviously, one can see that only one of the literal can be false, i.e., either $(min^{ctl}_{\rho_1} \leq max^{ctl}_{\rho_2})$ or $(min^{ctl}_{\rho_2} \leq max^{ctl}_{\rho_1})$ is false. For the sake of simplicity, we denote M^{ctl} a worst-case scenario of $\mathcal{M}$. The left network of Figure 3d forms a non-WC cycle. The controllable bounds are $\{30, 20\}$ on (A-B) that forms a path ρ_1, $\{10, 20\}$ on (C-B) and $\{15, 10\}$ on (A-C) that together form a path ρ_2. We have $\Pi^{ctl}_{\rho_1} = \{30, 20\}$ and $\Pi^{ctl}_{\rho_2} = \{25, 30\}$, which does not satisfy $min^{ctl}_{\rho_2} \leq max^{ctl}_{\rho_1}$.

6 The WC-Checking algorithm

6.1 Description of the algorithm

In this section, we present the new WC-checking algorithm for an STNU $\mathcal{X}$, which comprises two parts: the first finds the cycles from a divergent time-point, and the second checks those cycles. It is based on the following basic structures:

- A path ρ is divided into two **projection paths** ρ_{min} and ρ_{max} where only the minimal (ρ_{min}) or maximal (ρ_{max}) controllable bounds are computed: $\Pi^{ctl}_{\rho_{max}} = max^{ctl}_{\rho}$ and $\Pi^{ctl}_{\rho_{min}} = min^{ctl}_{\rho}$. Given $\eta = \{min, max\}$, a projection path is of the form $\rho_\eta = \langle \eta^{ctl}_{\rho}, C_{\rho_\eta}, \mathcal{V}_{\rho_\eta}\rangle$ such that
  - η^{ctl}_{ρ} is the controllable bound of ρ_η (max^{ctl}_{ρ} or min^{ctl}_{ρ});
  - C_{ρ_η} is the set of contingent constraints of ρ_η ($C_{\rho_\eta} \subseteq C$);
  - $\mathcal{V}_{\rho_\eta}$ the set of time-points of ρ_η ($\mathcal{V}_p \subseteq \mathcal{V}$).

  One can notice that ρ_{min} and ρ_{max} represent the two worst-case scenarios of ρ.
- $\mathcal{P}(v_d)$ is the set of **projection paths** $\mathcal{P}(v_d) = \{\rho_{\eta_1}, \dots, \rho_{\eta_m}\}$ from the divergent time-point v_d.

- the **minimal divergent cycles** $D_{min}(v_d)$ is a mapping of convergent time points $v_c \in \mathcal{V}_{cv}$ to a set of projection paths $(\mathcal{P}^{min}_{v_c})$ that converge on v_c from v_d such that each of them η = min (ρ_{min}): $\forall \rho_\eta \in \mathcal{P}^{min}_{v_c}, \eta^{ctl}_\rho = min^{ctl}_\rho$.
- the **maximal divergent cycles** $D_{max}(v_d)$ is similar as $D_{min}(v_d)$ but each projection path in $\mathcal{P}^{max}_{v_c}$, η = max (ρ_{max}): $\forall \rho_\eta \in \mathcal{P}^{max}_{v_c}, \eta^{ctl}_\rho = max^{ctl}_\rho$.

We introduce in Algorithm 1 the *findDivergentCycles* algorithm in charge of finding the cycles from 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 is common in qualitative temporal networks [6]: it is possible to define a partial order of all time-points with regard to the precedence relation; $rank(v_z) = 0$, then for all v_i such that $v_z \preceq v_i \in E \cup C$ and there is no v_j such that $v_z \preceq v_j \in E \cup C$ 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_z (rank 0). This enables us to avoid any time-point v_i with a lower rank than the current divergent time-point v_d. Figures 3a to 3c highlight only the edges considered by the forward search.
- To distinguish 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 projection paths ρ_η and ρ'_η such that $C_{p_\eta} = C_{p'_\eta} = \{\emptyset\}$ (only requirement constraints) ρ_η is stricter than ρ'_η if η = min and $min^{ctl}_\rho > min^{ctl}_{\rho'}$ (respectively, η = max and $max^{ctl}_\rho < max^{ctl}_{\rho'}$). Hence, it's useless to consider further ρ'_η as ρ_η is a stricter projection path, and only ρ_η is kept in $D_{min}(v_d)$ or $D_{max}(v_d)$ depending on the computed controllable bound (η). This also holds for a path ρ'_η such that $C_{\rho'_\eta} \neq \{\emptyset\}$ (with contingent constraints). However, when C_{ρ_η} and $C_{\rho'_\eta}$ are not empty, applying these rules is impossible as it might result in removing an inconsistent cycle in the graph. Suppose we have the minimal controllable bounds of ρ and ρ' (min^{ctl}_ρ and $min^{ctl}_{\rho'}$) and the maximal controllable bounds of a path ρ'' ($max^{ctl}_{\rho''}$) such that the pair $\langle \rho', \rho'' \rangle$ forms a cycle M^{ctl}. Then, if ρ_{min} is stricter than ρ'_{min} and ρ'_{min} is not kept, M^{ctl} will never be checked likewise for the WC of $\mathcal{X}$. Therefore, both ρ_{min} and ρ'_{min} must be kept in $D_{min}(v_d)$. [1] We illustrate such case in Figure 2.

Lines 1 to 3 initialize the maps $D_{max}(v_d)$ and $D_{min}(v_d)$, and the set of paths $\mathcal{P}(v_d)$. Then, lines 5-16 propagate the paths in $\mathcal{P}(v_d)$ to find and keep all stricter paths of v_d in $D_{max}(v_d)$ until $\mathcal{P}(v_d) = \{\emptyset\}$. In fact, in line 14, we also update $\mathcal{P}(v_d)$ and $\mathcal{P}_{v_j}$ by removing the paths that are not stricter anymore. A second forward search is done for $D_{min}(v_d)$ where $\mathcal{P}(v_d)$ 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 (ρ_{min}, ρ_{max}) that converge on v_c and form a cycle M^{ctl} where $\mathcal{V}_{\rho_{min}} \cap \mathcal{V}_{\rho_{max}} = \{v_d, v_c\}$.

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 the time-points (line 2), and find and check the cycles of each divergent time-point in $\mathcal{V}_{dv}$.

[1] 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.

Algorithm 1 findDivergentCycles algorithm.

```
Input: $v_d$:(time-point), $\Pi_{\mathcal{X}}^{ctl}$: (graph), rank: map
Output: Boolean
1  $D_{min}(v_d) = D_{max}(v_d) = \{\}$
2  $\mathcal{P}(v_d) = [\langle 0, [], [v_d] \rangle]$
3  A first forward search for $D_{max}$
4  while $\mathcal{P}(v_d)$ not empty do
5      $\rho_{max} = P(v_d)[0]$    $\rho_{max}$ is removed in $\mathcal{P}(v_d)$
6      for each child $v_j$ of $v_m \in \mathcal{V}_{\rho_{max}}$ with rank($v_j$) $\geq$ rank($v_d$) and $v_j \notin \mathcal{V}_{\rho_{max}}$ do
7          $\rho_{max}$ = propagateMaxPath($\Pi_{\mathcal{X}}^{ctl}$, $\rho_{max}$, $max_{mj}^{ctl}$)
8          if $v_j$ is a convergent time point ($v_j \in \mathcal{V}_{cv}$) then
9              if $v_j$ not in $D_{max}(v_d)$ then
10                 add $v_j \rightarrow [\rho_{max}]$ in $D_{max}(v_d)$
11             else
12                 if $\rho_{max}$ is a restrictive path in $\mathcal{P}_{v_j}^{max}$ then
13                     add $\rho_{max}$ to $\mathcal{P}_{v_j}^{max}$ and to $\mathcal{P}(v_d)$
14         else
15             add $\rho_{max}$ to $\mathcal{P}(v_d)$
16 A second forward search for $D_{min}(v_d)$ with $\rho_{min}$
17 return checkCycles($D_{max}(v_d)$, $D_{min}(v_d)$)
```

Algorithm 2 checkCycles algorithm.

```
Input: $D_{max}(v_d)$, $D_{min}(v_d)$
Output: Boolean
1  for each $v_c \rightarrow \mathcal{P}_{v_c}^{min}$ in $D_{min}(v_d)$ do
2      for each $\rho_{min}$ in $\mathcal{P}_{v_c}^{min}$ do
3          for each $\rho_{max}$ in $\mathcal{P}_{v_c}^{max}$ in $D_{max}(v_d)$ do
4              if ($\rho_{min}$, $\rho_{max}$) is of the form $M^{ctl}$ then
5                  if $min_{\rho}^{ctl} > max_{\rho}^{ctl}$ then
6                      return False    Or the cycle
7  return True
```

Algorithm 3 WC-Checking algorithm.

```
Input: $\mathcal{X}$: STNU($\mathcal{V}$,E,C)
Output: Boolean
1  $\Pi_{\mathcal{X}}^{ctl}$ = getDistanceGraph($\mathcal{X}$)
2  rank = orderFromRank($\mathcal{X}$)
3  for each $v_d$ in $\mathcal{V}_{dv}$ do
4      if findDivergentCycles($v_d$, $\Pi_{\mathcal{X}}^{ctl}$, rank ) == False then
5          return False  Or non-WC cycles of $v_d$
6  return True    Or all non-WC cycles
```

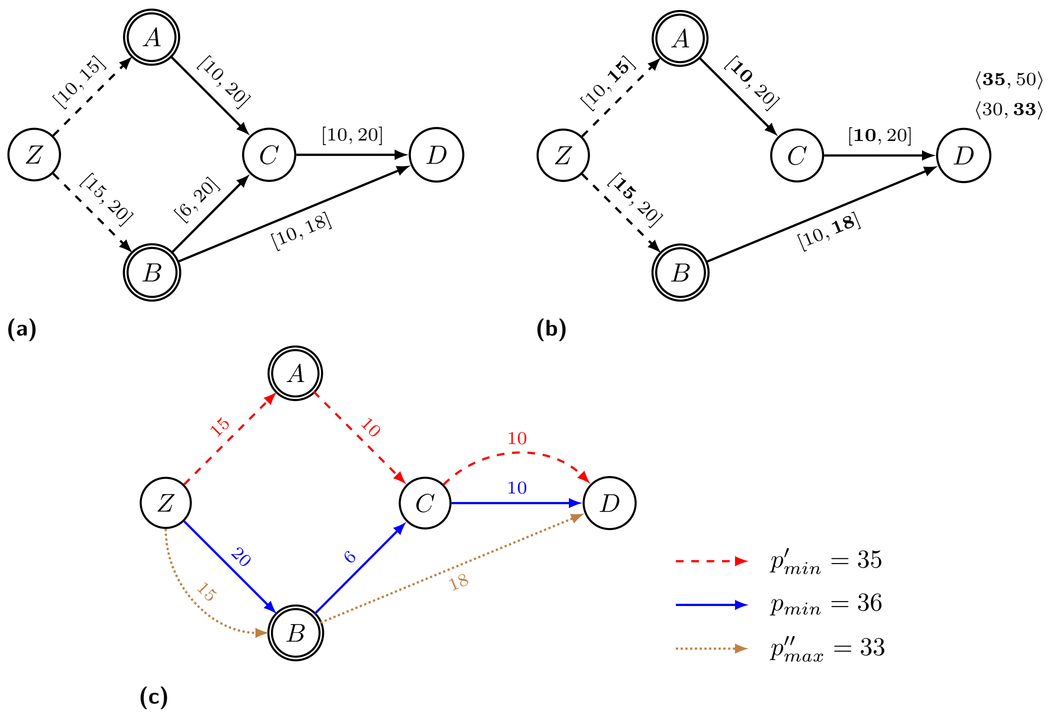


Figure 2 This figure illustrates the special case of the pruning rules when C_{ρ_η} and $C_{\rho'_\eta}$ are not empty. Figure a) shows a non-Weakly controllable STNU, whereas Figure b) shows its only non-WC cycle. Figure c) highlights the computed projection paths p_{min}, p'_{min}, and p''_{max} of the given example. One can see that if p'_{min} is not kept in $D_{min}(v_d)$, the inconsistent cycle will never be checked as the pair $\langle p_{min}, p''_{max} \rangle$ do not form a cycle M^{ctl}. Hence, such pruning rules cannot be applied when C_{ρ_η} and $C_{\rho'_\eta}$ are not empty.

We show the execution of our algorithm for Divergent time-point A in Figure 3 using $A \rightarrow C \rightarrow B \rightarrow D$ as the order for the forward searches. We highlight the search and the paths (min and max) forming the non-Weakly controllable cycle. Please note that we simplified the example by not showing how D_{min} and D_{max} are incrementally changed.

6.2 Features and Complexity

The algorithm presented in the previous section returns the set of negative cycles of a non-Weakly controllable STNU (see Algorithms 2, 3), which is important for explainability, i.e., necessary for the repair problem. Moreover, divergent time-points are independent, which makes parallelization possible. In addition, the usual pseudo-controllability step from Morris [10] is not required for constraint bounds with finite values ($l_{ij} \neq -\infty$ and $u_{ij} \neq +\infty$). Thus, an incremental execution is possible as divergent time-points are independent. Indeed, when adding new constraints, it's not necessary to recompute the minimal network; hence, checking only the cycles of divergent time points of the same rank or lesser (topological ordering) is enough to guarantee WC. Still, it is not optimal as unnecessary cycles might be checked. The drawback of the algorithm is that the minimal network 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 not better than the original algorithm

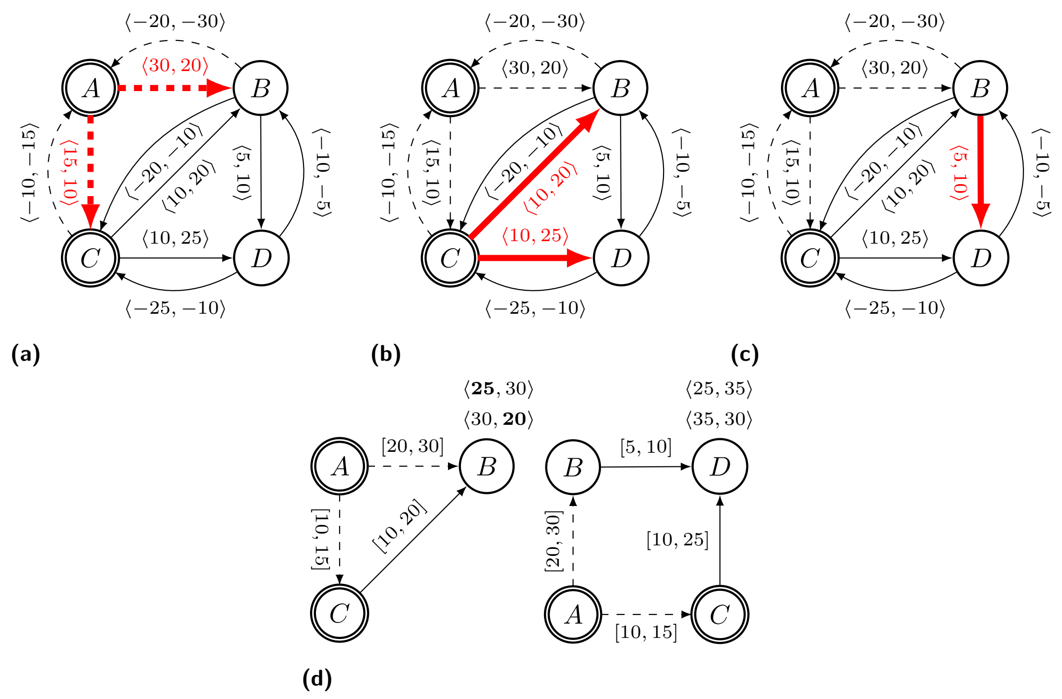


Figure 3 This Figure shows, in a simplified manner, a running example of Algorithm 1 with divergent time-point A. We highlight the edges taken at each step according to line 7. Figures 3a to 3c show the search, while Figure 3d shows the cycles to check for A, with the left one being not Weakly controllable. After step 3, D_{min} and D_{max} contain all the restrictive paths (only those that need to be kept) with, in a simplified manner, $D_{min} = \{C : \langle 15, AC\rangle, B : [\langle 20, AB\rangle, \langle \mathbf{25}, \mathbf{ACB}\rangle], D : [\langle 15, ACD\rangle, \langle 35, ABD\rangle, \langle 30, ACBD\rangle]\}$ and $D_{max} = \{C : \langle 10, AC\rangle, B : [\langle \mathbf{20}, \mathbf{AB}\rangle, \langle 30, ACB\rangle], D : [\langle 25, ACD\rangle, \langle 30, ABD\rangle, \langle 40, ACBD\rangle]\}$. We highlight the paths of the non-Weakly controllable cycle.

$(2^{|C|})$. However, our interest lies in realistic graphs where the sparsity of the graph is low by restricting these parameters. Thus, the next section compares our algorithm (new_WC) with the original one (old_WC) using the Floyd-Warshall algorithm (APSP) as a time metric only to see how close they are to a polynomial behavior when parameters are restricted enough.

7 Experiments

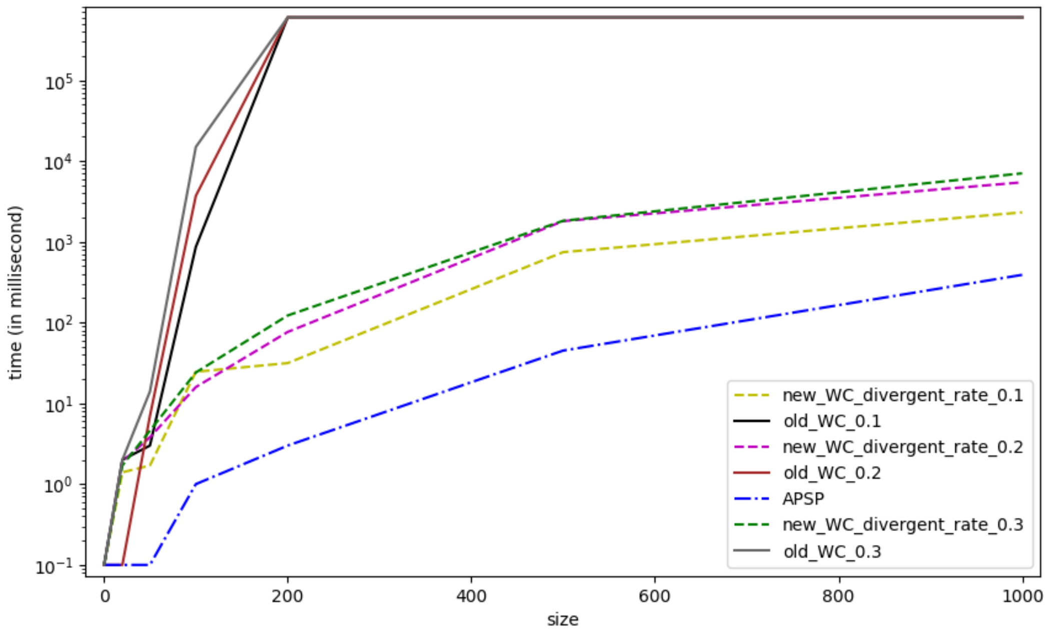
To empirically test the effectiveness of the proposed algorithm, we consider the execution time as the execution of all computations and not after finding an inconsistency as existing checking algorithms do. The benchmark comes from a random generator we implemented that can generate sparse STNUs. It creates an STNU in the form of a complete directed acyclic graph (DAG), then randomly removes several edges depending on parameters: the number of time points n, the percentage of divergent time points r_d, the maximum number of their successors n_c, and the percentage of contingent constraints 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.

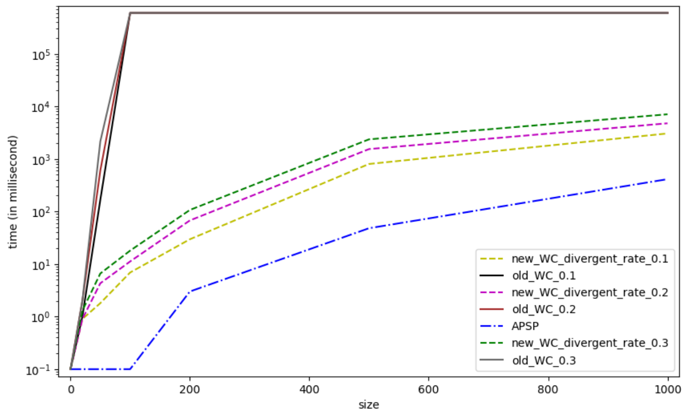
We experiment under different settings: n $= \{20, 50, 100, 200, 500, 1000\}$, $r_d = \{0.1, 0.2, 0.3\}$ meaning 10 to 30% of divergent time-points, $r_c = \{0.2, 0.3\}$, and $n_c = 3$. For each combination of parameters, we generate 20 STNUs and compute the average execution time. We show in Figure 4b that, in general, our algorithm clearly outperforms the *old-WC* algorithm and has a behavior slightly worse than the APSP algorithm up to 20% of contingent constraints. This shows that the parameters were bounded enough to have a polynomial-like behavior. However, beyond this threshold, our algorithm starts to show its limit. This shows the sensitivity of our algorithm to the parameters (see Figure 4c). In addition, we observe from the experimentation that the position of contingents can impact the number of cycles to check. The closer to v_0 contingents are, the higher the number of cycles to check. Such a case is shown in Figure 4a where the dotted line for the case of 10% of divergent time-points (new_WC) overlaps the other two (20 and 30 %).

8 Conclusion

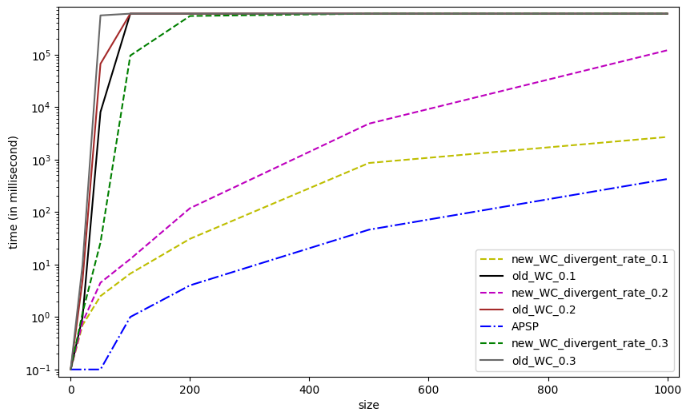
This paper introduced a novel approach for checking the WC of an STNU by checking the consistency of its elementary cycles. Interesting features of our algorithm to consider further are as follows: it can identify the constraints causing the uncontrollability, and it can be executed in an incremental 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.



(a)



(b)



(c)

Figure 4 Experimentation with 10% (a), 20% (b), and 30% (c) of contingent constraints.

References

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. doi:10.1613/JAIR.1.13065.

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

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

6 Malik Ghallab and A. Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 1297–1303. Morgan Kaufmann, 1989. URL: http://ijcai.org/Proceedings/89-2/Papers/072.pdf.

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. doi:10.1002/NET.3230250306.

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. A structural characterization of temporal dynamic controllability. In *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, volume 4204 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 2006. doi:10.1007/11889205_28.

11 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. doi:10.1007/978-3-319-07046-9_33.

12 Paul H. Morris and Nicola Muscettola. Managing temporal uncertainty through waypoint controllability. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1253–1258. Morgan Kaufmann, 1999. URL: http://ijcai.org/Proceedings/99-2/Papers/083.pdf.

13 Ajdin Sumic, Alessandro Cimatti, Andrea Micheli, and Thierry Vidal. SMT-based repair of disjunctive temporal networks with uncertainty: Strong and weak controllability. In *Proceedings of the The 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2024)*, 2024.

14 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. doi:10.1080/095281399146607.