Centralized and Distributed Approaches for Restoring the Weak Controllability of Multi-Agent Interdependent STNUs

Ajdin Sumic[1] , **Gauthier Picard**[1] , **Roberto Posenato**[2] ,
Thierry Vidal[3] , **Carlo Combi**[2] and **Frédéric Maris**[4]
[1]DTIS, ONERA, Université de Toulouse, France
[2]University of Verona, Italy
[3]Université de Technologie Tarbes Occitanie Pyrénées, France
[4]Institut de Recherche en Informatique de Toulouse, France
{aidin.sumic, gauthier.picard}@onera.fr, {roberto.posenato, carlo.combi}@univr.it,
thierry.vidal@uttop.fr, frederic.maris@irit.fr

Abstract

Temporal planning and scheduling often rely on constraint models to reason about activity durations. Simple Temporal Networks with Uncertainty (STNUs) address cases where some *contingent* durations are outside the control of the executing agent. The Multiple Interdependent STNU (MISTNU) model extends this setting to multi-agent systems by representing shared activities whose duration is decided by one agent and imposed on others. While controllability properties for MISTNUs have been defined, existing repair methods rely on a centralized SMT-based brute-force approach. This paper introduces a linear-constraint characterization of all negative cycles responsible for uncontrollability. Based on this formulation, we propose both a more efficient centralized linear-programming repair method and a distributed constraint reasoning approach that treats inconsistent cycles as inter-agent constraints. We experimentally compare several Distributed Constraint Optimization Problem (DCOP) solvers against the centralized baseline in terms of efficiency and repair quality.

1 Introduction and Related Work

In multi-agent temporal planning problems, agents must first collaboratively construct individual plans through task allocation and planning techniques, possibly involving negotiation, and then check and coordinate the temporal executability of their plans. In this paper, we focus exclusively on the latter stage and on the specific case where the task owner controls task durations. At the same time, other compliant agents may depend on those durations and may learn them only just before or during execution. Let us consider an example from the healthcare domain: *the radiology department waits for a patient's surgery, performed by the emergency department, to finish. The duration of the surgery is therefore uncertain for the former, since it is set by the latter and may be communicated only in accordance with the surgery department's policy.* Such compliant agents must check the executability of their plans, but may also try to *repair* a non-executable plan by negotiating shared constraints.

Simple Temporal Networks (STNs) [Dechter *et al.*, 1991] are a well-known framework for managing temporal duration and checking temporal consistency of plans. They have been extended to STNs with Uncertainty (STNU) [Vidal and Fargier, 1999] to represent uncertain (*contingent*) durations, redefining consistency as *controllability*: an STNU is controllable if a strategy exists for executing the plan, whatever the values taken by the contingent durations. Three levels of controllability exist, depending on when the contingents' durations will be known: just before execution (*Weak*), during execution (*Dynamic*), or never (*Strong*).

Extensions to multi-agent settings were first addressed in Multi-agent STNU (MaSTNU [Casanova *et al.*, 2016]) or Disjunctive TNU [Bhargava and Williams, 2019] (MaDTNU) models that consider common exogenous uncertainty, i.e., contingent constraints still come from *Nature* and affect all agents equally. Some approaches propose decoupling algorithms for Dynamic Controllability that ensure each agent is controllable [Casanova *et al.*, 2016] [Zhang and Williams, 2021]. Another centralized approach with common exogenous contingent constraints introduces the concept of a *contract* to represent interdependent activities across agents [Lubas and Eder, 2023]. These contracts are similar to classical inter-agent constraints and are compiled into a single STNU to verify Dynamic Controllability. Overall, these models enforce central controllability checking, which is well-suited to multi-agent planning problems in which a common task is divided among different participants.

More recently, the Multi-Agent Interdependent STNUs (MISTNU) model introduced a semantically distinct notion of *contract*, corresponding to a shared constraint being controllable for one agent but contingent for others [Sumic *et al.*, 2024b]. It distinguishes between exogenous and inter-agent contingencies, preserving privacy, and defining clear execution and observation semantics at various controllability levels. That model fits semi-decentralized multi-agent planning, where each agent plans its own course of action while coordinating through mutual commitments. The au-

thors define the distributed controllability-checking problem for MISTNUs and propose a centralized *repair* method to restore controllability when needed, by negotiating contract durations using Satisfiability Modulo Theories (SMT) [Barrett and Tinelli, 2018].

Meanwhile, new algorithms have emerged for detecting the causes of Weak and Dynamic uncontrollability of STNUs in the form of inconsistent negative cycles [Sumic and Vidal, 2024; Hunsberger and Posenato, 2022]. In this paper, we show that such "informed" algorithms can improve the current "blind" centralized repair by specifically targeting those cycles. Our first contribution is to translate negative cycles into linear constraints, focusing on the *Weak Controllability* repair problem, and to develop a centralized linear-programming approach. Experiments show that, although the number of negative cycles is theoretically exponential, they can be repaired efficiently.

Our second contribution proposes a distributed approach to the same problem, still leveraging the linear constraints that encode the negative cycles. The problem is addressed from a Distributed Constraint Reasoning perspective [Yokoo *et al.*, 1998] and modeled as a Distributed Constraint Optimization Problem [Modi *et al.*, 2005] that can be solved by any distributed method. We notably assess the performance of optimal algorithms ADOPT, AFB, DPOP, and SyncBB. In such approaches, agents implement a coordination protocol to reach an agreement over the contract bounds.

The paper provides the necessary background on STNUs and MISTNUs in Section 2, then presents our contributions in Sections 3 to 4, assessing them through comprehensive experiments in Section 5, and concludes with a few prospects.

2 Background

2.1 Single Agent Temporal Models

A Simple Temporal Network (STN) is a tuple $\langle \mathcal{V}, E \rangle$, where $\mathcal{V}$ is a set of real-value variables called timepoints $\{v_1, \ldots, v_n\}$, and E is a set of temporal constraints between timepoints called *requirements* [Dechter *et al.*, 1991]. Specifically, each requirement $e_k \in E$ is of the form $v_j - v_i \in [l, u]$, with $v_i, v_j \in \mathcal{V}$, $l \in \mathbb{R} \cup \{-\infty\}$, and $u \in \mathbb{R} \cup \{+\infty\}$; l and u represent the minimal and maximal possible temporal distance between v_i and v_j. A requirement is represented as an edge $v_i \xrightarrow{[l,u]} v_j$. A reference timepoint v_z is generally included in $\mathcal{V}$ as a fixed temporal anchor assumed to be the first executed timepoint of the network, i.e., v_z will take a value such that each implicit constraint $v_z \leq v_i$ for each $v_i \in \mathcal{V}$ is satisfied.

An STN is **satisfiable** if a timepoint assignment (schedule) exists that satisfies all the constraints. We say that a controller executes an STN when it schedules its timepoints.

An STN with Uncertainty (STNU) is an extension of STNs in which one distinguishes a subset of constraints whose values (duration) are decided by external entities (i.e., *Nature*) and the controller can only observe them [Vidal and Fargier, 1999]. An STNU $\mathcal{X} = \langle \mathcal{V}, E, C \rangle$ is a tuple where $\mathcal{V}$ is now divided into controllable ($\mathcal{V}_c$) and uncontrollable ($\mathcal{V}_u$) timepoints. C is a set of (uncontrollable) contingent links c_k each of the form $v_j - v_i \in [l, u]$, where $v_i \in \mathcal{V}_c$, $v_j \in \mathcal{V}_u$, and $0 < l \leq u$. The value (*duration*) $\omega_k = v_j - v_i$ is set by an external entity before or during the execution. Once v_i is executed, the value of v_j is $v_i + \omega_k$. Any c_k is depicted as $v_i \overset{[l,u]}{\dashrightarrow} v_j$. A **schedule** for an STNU is a mapping $\delta \colon \mathcal{V} \to \mathbb{R}$ from timepoints to real values.

Definition 1 (Situation, Projection and Solution)**.** *Given an STNU $\mathcal{X} = \langle \mathcal{V}, E, C \rangle$, the **situations** of $\mathcal{X}$ are the set of tuples Ω defined as the Cartesian product of contingent domains: $\Omega = \times_{c_k \in C}[l_k, u_k]$. Each situation $\omega = \{\omega_1, \ldots, \omega_n\} \in \Omega$ represents one possible complete set of values for the durations of the contingent links of $\mathcal{X}$.*
*A **projection** $\mathcal{X}^\omega = (\mathcal{V}, E \cup C^\omega)$ of $\mathcal{X}$ is an STN, where C^ω replaces each contingent link c_k with the rigid requirement induced by ω_k. A **solution** of $\mathcal{X}^\omega$ is a schedule δ_ω satisfying all the constraints.*

Intuitively, a projection replaces each contingent link with a rigid requirement, i.e., a constraint reduced to a singleton.

In STNUs, consistency needs to be redefined: a network is now said to be *controllable* if there exists a schedule that satisfies all requirement constraints in any possible projection.

Three levels of controllability exist: *Strong Controllability (SC)* assumes that a single common schedule δ satisfies all constraints in all situations, which is relevant when the durations of external events cannot be observed (conformant planning). *Weak Controllability* (WC) assumes there is one consistent schedule for each situation, which is relevant when all contingent durations will be known just before execution (oracle). *Dynamic Controllability (DC)* assumes the schedule values assignment depends on past observations only, regardless of the contingent durations still to be observed. In this paper, we only recall the formal definition of WC, which is our focus.

Definition 2 (Decision and Observation)**.** *$\forall v_i \in \mathcal{V}_c$, $dec(v_i)$ is the instant at which $\delta(v_i)$ is **decided** by the controller. $\forall \omega_k \in C$, $obs(\omega_k)$ is the instant at which ω_k is **observed** by the controller.*

Definition 3 (Weak Controllability (WC))**.** *An STNU $\mathcal{X}$ is **weakly controllable** iff $\forall \omega \in \Omega, \exists \delta_\omega$, which is a solution of $\mathcal{X}_\omega$. Execution semantics: $\forall \omega_k \in \omega$, $obs(\omega_k) = v_z$, and the decision policy is free: $\forall v_i \in \mathcal{V}_c$, $dec(v_i) \leq v_i$.*

DC and WC checking rely on algorithms to check if there are negative cycles in the *distance graph* of STNUs [Hunsberger and Posenato, 2022; Sumic *et al.*, 2024a]. The distance graph of an STNU is a graph $\mathcal{X}_D = \langle \mathcal{V}, \mathcal{E} \rangle$ where a requirement constraint in E is represented through two edges $v_i \xrightarrow{u} v_j$ and $v_j \xrightarrow{-l} v_i$ in $\mathcal{E}$. A contingent link in C is represented through four edges: two as above, a lower-case (LC) edge $v_i \xrightarrow{j:l} v_j$, and an upper-case (UC) edge $v_j \xrightarrow{J:-u} v_i$. The LC $v_i \xrightarrow{j:l} v_j$ (*resp.* UC $v_j \xrightarrow{J:-u} v_i$) represents the uncontrollable possibility that the duration $v_j - v_i$ might take its minimum value l (*resp.* maximum value u).

Definition 4 (Inconsistent Cycle)**.** *Given a distance graph $\mathcal{X}_\mathcal{D}$, a* **cycle** $\varrho = (v_i, \alpha_{ij}, v_j, \ldots, v_k, \alpha_{km}, v_m)$ *is a sequence of timepoints and edge values s.t $v_i \equiv v_m$ and where each $\alpha_{kk'}$ is the (unlabeled or LC or UC) value of the considered edge connecting v_k to $v_{k'}$. We denote by $\alpha(\varrho)$ the* length

*of a cycle, i.e., the sum of the values of the edges composing the path, ignoring possible labels. A cycle ϱ is **negative/inconsistent** iff $\alpha(\varrho) < 0$. We denote by $\mathcal{M} = \{\varrho_1, \ldots, \varrho_n\}$ the set of all inconsistent cycles of an STNU.*

2.2 Multi-Agent Interdependent STNU

STNUs have been extended to a multi-agent context, considering that the duration of a link that is controllable for one agent may be observed as contingent by other agents [Sumic *et al.*, 2024b]. Such shared and duplicated constraints are controllable for the agent that makes the decision, but are contingent for an agent that depends on such decisions. To express such semantics, the authors first define a *contract* p as a label that is associated with a range of durations that can be assigned to distinct binary constraints in different STNUs.

Definition 5 (cSTNU)**.** *A Contracting STNU (cSTNU) is an STNU extended with links being labeled by contracts. A cSTNU is a tuple $\mathcal{S} = \langle \mathcal{V}, R, W, E, C, O \rangle$ such that:*

- *$\mathcal{V}$ is defined as in STNUs.*
- *R and W are sets of owned (W) and observed (R) contracts, such that $R \cap W = \emptyset$.*
- *E is the set of **local** requirement constraints, which are not related to contracts.*
- *C is a set of* labeled *contingent links of the form $v_i \xrightarrow{p} v_j$ where $p \in R$, corresponding to the **observed** contracts.*
- *O is a set of owned contract links of the form either $v_i \xrightarrow{p} v_j$ or $v_i \xrightarrow{p} v_j$, where $p \in W$.*

In other words, C is now related to observed contracts, which are necessarily contingent. While constraints corresponding to owned contracts are now in O, they will retain the same status, either as a requirement or a contingent link. At checking time, they will be considered as contingent since the interval of possible values is a commitment between a writer and the readers, that even the owner is not supposed to shrink while others check their controllability with respect to them: then requirements are in E and contingents in $C \cup O$; while at execution time they may be considered as controllable so that the owner can modify the bounds or set an effective value: then requirements are in $E \cup O$ and contingents in C.

Although the model can still support exogenous non-repairable contingent constraints, in this paper, we assume that all contracts have an owner, i.e., each constraint in C is issued by another agent that has the same contract in its own set O.

Alongside cSTNUs, they define a new multi-agent model called *Multi-agent Interdependent STNU* composed of a set of agents, each with its cSTNU, and a reification function $\mathcal{B}$ that, given a contract p, returns the range $[l_p, u_p]$ of possible durations for p. For any cSTNU $\mathcal{S}$, we call *reification* of $\mathcal{S}$ the application of $\mathcal{B}$ to its contracts, replacing all labels with the interval delimited by $\mathcal{B}$, i.e., $[l_p, u_p]$, returning an STNU.

Definition 6 (MISTNU)**.** *A MISTNU is a tuple $\mathcal{G} = \langle A, \Sigma, \mathcal{B} \rangle$ such that:*

- *A is a set of agents $\{a_1, a_2, \ldots, a_n\}$;*
- *Σ is a set of cSTNUs $\mathcal{S}_a = \langle \mathcal{V}_a, R_a, W_a, E_a, C_a, O_a \rangle$, one for each $a \in A$, such that*
  - *$\forall a \in A, P_a = R_a \cup W_a$ is the set of contracts of agent a, and $v_z \in \mathcal{V}_a$ is the mutual reference timepoint for the sake of synchronization.*
  - *for every pair of agents $a, b \in A$, $W_a \cap W_b = \emptyset$*
- *$\mathcal{B}\colon P \to \mathbb{R}^2$ is the global reification function for all the contracts $P = \bigcup_{a \in A} P_a$.*

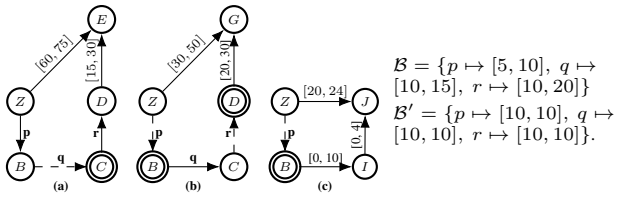

Figure 1: A MISTNU composed of three agents *a*, *b*, and *c* and their cSTNUs. Contracts p and r are owned by *a*, and q by *b*. Z is the reference timepoint v_z. Doubly circled nodes are contingent. $\mathcal{B}$ and $\mathcal{B}$' are the reification functions before and after repair, respectively.

Example 1. *Consider a radiologist (agent a), a nurse (agent b), and a doctor (agent c). The radiologist performs an X-ray on the first patient, represented by contract p, whose duration is observed by the nurse and the doctor to be contingent: the nurse waits for the X-ray before escorting the patient, while the doctor can analyze the results only after its completion. The nurse then escorts the current patient and brings a second patient to the radiologist, represented by contract q, which is owned by the nurse but observed by the radiologist. Finally, contract r represents the second X-ray, along with the following dependencies involving the nurse and the radiologist. After this second X-ray, the radiologist must clean and inspect the equipment within the required time window. Figure 1 illustrates the corresponding MISTNU, including contract ranges, deadline constraints, and additional local constraints.*

Although the model could be fully decentralized, the contract bounds are defined by $\mathcal{B}$, which serves as a common, blackboard-like external and public structure [Botti *et al.*, 1995], making MISTNU a *semi-decentralized* model in which only information about the shared contracts is public.

The MISTNU model is useful for assessing the controllability of each agent in a distributed manner, depending on when the contract duration is shared. We tackle the case of WC, where such communication happens before execution and has proven highly relevant in applications [Sumic and Vidal, 2024] when, e.g., some resource availability is not known in advance by an agent but will be set together with related task durations at the start of the day, which is, for instance, often met in healthcare services.

Execution semantics. In the WC setting, a situation is a complete assignment of durations to the contracts reified as contingent links. This assignment is fixed before execution and communicated to all agents at the initial reference timepoint v_z. Thus, situations are not generated by agents' scheduling decisions: all agents react to the same global situation. The MISTNU WC condition, therefore, checks whether, for every possible global assignment of contract durations, each reified local STNU admits a feasible schedule.

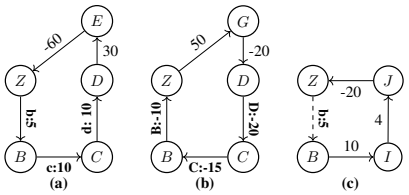

Figure 2: Inconsistent cycles of agent STNUs. Agent *a* STNU is not WC as there is a cycle of length $\alpha = -5$ when the contracts take their lower bound. The same applies to agent *b* when the contracts take their upper bound and to agent *c* when *p* takes its lower bound.

Definition 7 (MISTNU WC)**.** *The Weak Controllability of a MISTNU $\mathcal{G} = \langle A, \Sigma, \mathcal{B} \rangle$ is defined as $\forall \mathcal{S}_a \in \Sigma, \mathcal{S}_a^{\mathcal{G}}$ is WC, where $\mathcal{S}_a^{\mathcal{G}}$ is the reification of $\mathcal{S}_a$ to an STNU using $\mathcal{B}$.*

When the MISTNU is not controllable, i.e., at least one cSTNU is not controllable, the MISTNU may be repaired by shrinking the contracts' bounds as defined as follows for WC:

Definition 8 (Weak Repair)**.** *Given a MISTNU $\mathcal{G} = (A, \Sigma, \mathcal{B})$ such that, for some $a \in A$, $\mathcal{S}_a^{\mathcal{G}}$ is not WC, the **Weak Repair Problem** consists in finding a new reification function $\mathcal{B}'$ for a MISTNU $\mathcal{G}' = (A, \Sigma, \mathcal{B}')$ such that:*

- *$\forall p \in P$, if $\mathcal{B}(p) = [l_p, u_p]$ and $\mathcal{B}'(p) = [l'_p, u'_p]$, then $l_p \leq l'_p \leq u'_p \leq u_p$.*
- *$\mathcal{G}'$ is WC, i.e., $\forall a \in A$, $\mathcal{S}_a^{\mathcal{G}'}$ is WC.*

In [Sumic *et al.*, 2024b], the authors used a standard exponential WC checking method [Vidal and Fargier, 1999] and, when it fails, proposed a centralized SMT-based repair algorithm that encodes all the projections of each agent network and searches for the minimal reduction of the contracts that satisfy these projections. The MISTNU in Figure 1 with function $\mathcal{B}$ is not WC, and we show that each agent STNU contains a negative cycle, as depicted in Figure 2. The function $\mathcal{B}'$ in Figure 1 provides new bounds that make the MISTNU WC.

A recent work proves that checking WC of MISTNU is co-NP-complete, while its (centralized) optimal repair problem has been determined to be in Σ_2^P [Maris *et al.*, 2025].

2.3 Distributed Constraint Optimization

The *Distributed Constraint Reasoning (DCR)* framework enables formal modeling of coordination and decision-making among multiple agents, each controlling variables, to collectively satisfy or optimize constraints defined over these variables. We are interested in the latter one, called Distributed Constraint Optimization Problems (DCOPs), to find optimal solutions, which is known to be NP-hard [Modi *et al.*, 2005].

Definition 9 (DCOP [Modi *et al.*, 2005])**.** *A* Distributed Constraint Optimization Problem (DCOP) *is formally represented by a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \mu \rangle$:*

- *$\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$ is a set of agents;*
- *$\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of discrete variables;*
- *$\mathcal{D} = \{d(x_1), \ldots, d(x_n)\}$ is a set of finite domains, such that variable x_i takes values in $d(x_i) = \{v_1^i, \ldots, v_k^i\}$;*
- *$\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of soft constraints, where f_i is a function that defines a cost $\in \mathbb{R} \cup \{\infty\}$ for each combination of values to the variables in its scope;*
- *$\mu \colon \mathcal{X} \to \mathcal{A}$ is a function that assigns the control of each variable to an agent.*

A solution to a DCOP is a complete assignment σ that minimizes a global objective function $F(\sigma)$ that aggregates the individual costs f_i's: $\sigma^* = \operatorname{argmin}_\sigma F(\sigma)$. The sum is generally used as an aggregation function: $\sigma^* = \operatorname{argmin}_\sigma \sum_{f_i \in \mathcal{F}} f_i$.

In a DCOP algorithm, each agent assigns values to its variables and communicates with peers to achieve a complete assignment, assuming that messages between agents may be delayed but are never lost. DCOP models can capture hard constraints by using infinite costs.

3 Centralized Weak Repair

In this section, we propose a centralized algorithm for the Weak Repair Problem that improves on the approach of [Sumic *et al.*, 2024b]. The previous method encodes all projections of the agents' networks and searches for a repair using SMT. Here, instead, we first use an informed WC-checking algorithm [Sumic and Vidal, 2024], which returns the causes of non-controllability as inconsistent cycles. We then encode only those cycles as linear constraints and solve the resulting linear-programming problem. We assume that a central agent can read the public contract bounds and impose a new reification function.

3.1 Weak Repair Through Inconsistent Cycles

The STNU Weak Repair Problem can be solved by reducing the bounds of contingent links [Sumic *et al.*, 2024a]. We prove that this can be obtained by considering only the contingent links present in inconsistent cycles of the non-WC network.

Theorem 1 (STNU inconsistent cycles repair)**.** *Let $\mathcal{X} = \langle \mathcal{V}, E, C \rangle$ be a non-WC STNU, and let $\mathcal{M} = \{\varrho_1, \ldots, \varrho_n\}$ be its set of inconsistent cycles involving contingent links. Repairing a non-WC STNU is equivalent to finding new contingent bounds (if any) for each cycle $\varrho_k \in \mathcal{M}$ such that ϱ'_k with new contingent bounds becomes consistent without creating any new inconsistent cycle.*

Proof. Let $\mathcal{X} = \langle \mathcal{V}, E, C \rangle$ be a non-WC STNU, and let $\mathcal{M} = \{\varrho_1, \ldots, \varrho_n\}$ be its set of inconsistent cycles. Since all relevant constraints must be satisfied in all possible situations, an inconsistent cycle indicates a situation in which the network is not executable. Shrinking contingent bounds removes possible situations and can only increase lower-bound contributions or decrease upper-bound contributions in the distance graph; hence, it cannot create new inconsistent cycles.

Thus, if all $\varrho_k \in \mathcal{M}$ are repaired, no inconsistent cycles remain. Hence, the newly formed STNU $\mathcal{X}' = \langle \mathcal{V}, E, C' \rangle$, with $C' = \{i \xrightarrow{[l', u']} j \mid i \xrightarrow{[l, u]} j \in C\}$ and $l \leq l' \leq u' \leq u$, satisfies Definition 3, since every projection $\mathcal{X}'^{\omega}$, with $\omega \in \Omega'$, admits a solution δ_ω [Sumic *et al.*, 2024a; Hunsberger and Posenato, 2022].

Conversely, assume that even reducing contingent links to singletons, i.e., $l' = u'$, is not enough to remove an inconsistent cycle. Then at least one projection $\mathcal{X}'^{\omega}$, with $\omega \in \Omega'$, does not satisfy Definition 3; hence, $\mathcal{X}'$ is not WC. □

Following Theorem 1, we extend it for MISTNUs by repairing the inconsistent cycles of each non-WC cSTNU.

Corollary 1 (Weak repair through inconsistent cycles)**.** *Given a non-WC MISTNU $\mathcal{G} = \langle A, \Sigma, \mathcal{B} \rangle$, its Weak Repair Problem can be solved by considering the inconsistent cycles of all reified agent STNUs. More precisely, for every $\varrho_k \in \bigcup_{a \in A} \mathcal{M}_a$, where $\mathcal{M}_a$ is the set of inconsistent cycles of $\mathcal{S}_a^{\mathcal{G}}$, one must find new contract bounds, if any, that make ϱ_k consistent as in Theorem 1.*

3.2 A Linear Encoding of Inconsistent Cycles

In this section, we show how to "solve" a set of inconsistent cycles by solving a linear programming (LP) problem. First of all, we show how to encode an inconsistent cycle ϱ in the distance graph $\mathcal{S}_a^{\mathcal{G}}$ of an agent a as a linear inequality to solve. Therefore, let us assume that $\varrho = (v_i, \alpha_{ij}, v_j, \ldots, v_k, \alpha_{km}, v_m)$ is an inconsistent cycle with length $\alpha(\varrho) = \sum_{ij} \alpha_{ij}$, where value labels are discarded, and that it contains at least one edge whose labeled value $\alpha_{kk'}$ is equal to $k'{:}l_p$ or $K'{:}{-}u_p$, where l_p/u_p is the current lower/upper bound associated with a contract p.

For each contract lower/upper bound l_p/u_p present in ϱ, we define a rational variable l^p/u^p. Then, in $\sum_{ij} \alpha_{ij}$, we replace each $\alpha_{k,k'} = l_p/u_p$ by the linear expression $l^p - l_p/u_p - u^p$ and remove any other unlabeled value. Then, the summation becomes a linear expression of the introduced variable. We call such an expression Υ^{ϱ}. More formally,

$$\Upsilon^{\varrho} = \sum_{v_l \xrightarrow{\alpha} v'_l \in \varrho} \begin{cases} l^p - l_p & \text{if } \alpha = v'_l{:}l_p \text{ (LC edge)} \\ u_p - u^p & \text{if } \alpha = V'_l{:}{-}u_p \text{ (UC edge)} \end{cases} \quad (1)$$

The inconsistent cycle is repaired by assigning values to the variables l^p and u^p such that the linear inequality $\Upsilon^{\varrho} \geq |\alpha(\varrho)|$ and the bounds $l_p \leq l^p \leq u^p \leq u_p$ are satisfied.

For example, the inconsistent cycle of agent a in Figure 2 is encoded as $(l^p - 5) + (l^q - 10) + (l^r - 10) \geq 5$, and a possible solution is $l^p = 10, l^q = 10$, and $l^r = 10$.

Now, it is possible to define how to solve the inconsistent cycles of an STNU and, in general, of an MISTNU. Given any STNU $\mathcal{S}_a^{\mathcal{G}}$ and the set of its inconsistent cycles $\mathcal{M}_a$, to repair each inconsistent cycle $\varrho_k \in \mathcal{M}_a$, it is sufficient to solve the linear programming problem defined by the set of linear constraints.

$$\Upsilon^{\mathcal{M}_a} = \begin{cases} \Upsilon^{\varrho_k} \geq |\alpha(\varrho_k)| & \forall \varrho_k \in \mathcal{M}_a \\ l_p \leq l^p \leq u^p \leq u_p & \forall \text{ contract } p \text{ involved} \end{cases} \quad (2)$$

To solve the Weak Repair Problem of an MISTNU, we must encode all the agents' inconsistent cycles using a common set of variables for the contract bounds. We denote by Ψ the set of constraints associated with the contracts' variables, and by Υ_{wc} the set of encodings of all the inconsistent cycles. More formally,

$$\Psi = \left\{ l_p \leq l^p \leq u^p \leq u_p \,\middle|\, p \in \bigcup_{a \in A} P_a, \mathcal{B}(p) = [l_p, u_p] \right\} \quad (3)$$

Algorithm 1: cycles_repair procedure

Input: A MISTNU $\mathcal{G} = \langle A, \Sigma, \mathcal{B} \rangle$
Output: $\mathcal{B}$ if $\mathcal{G}$ is WC, a new reification $\mathcal{B}'$ if $\mathcal{G}$ is not WC but repairable, $\emptyset$ otherwise.

```
all_cycles = []
foreach cSTNU S_a ∈ Σ do
    cycles = check_WC(S_a^G)
    all_cycles = all_cycles ∪ cycles
if all_cycles == ∅ then return B                    //G is WC
else
    B' = encode_cycles(all_cycles)   //Use eq. (1) and solve it by eq. (5)
    if B' == ∅ then return ∅                        //G is not reparable
    else return B'
```

$$\Upsilon_{wc} = \Psi \cup \left\{ \Upsilon^{\varrho_k} \geq |\alpha(\varrho_k)| \,\middle|\, \varrho_k \in \bigcup_{a \in A} \mathcal{M}_a \right\}. \quad (4)$$

Consequently, the Weak Repair Problem has a solution iff Υ_{wc} admits one. Among all possible solutions to the LP problem Υ_{wc}, we are interested in finding a solution that minimizes the modification of the current function $\mathcal{B}$. More formally, we solve the Weak Repair Problem by solving the following minimization linear programming problem:

$$\text{Min} \left(\sum_{\substack{p \in \bigcup_{a \in A} P_a \\ \mathcal{B}(p) = [l_p, u_p]}} \big((l^p - l_p) + (u_p - u^p) \big) \right) \quad \text{s.t. } \Upsilon_{wc} \quad (5)$$

Our methodology is summarized in Algorithm 1. Given a MISTNU $\mathcal{G}$, we extract all inconsistent cycles from each reified agent STNU $\mathcal{S}_a^{\mathcal{G}}$ using the approach in [Sumic and Vidal, 2024]. If no inconsistent cycle is found, $\mathcal{G}$ is WC. Otherwise, each inconsistent cycle is encoded using Equation (1), and the resulting system is solved using Equation (5). If no solution exists, $\mathcal{G}$ is deemed non-repairable.

4 Distributed Weak Repair Solving

In this section, we map the Weak Repair Problem for MISTNUs to a DCOP, which we call WRP-DCOP. As shown in Section 3, inconsistent cycles can be expressed as linear constraints over contract bounds. In the distributed setting, each agent locally checks the WC of its reified STNU and, if it is not WC, shares only the contract-bound information needed to negotiate a repair. These shared bounds define the DCOP variables, while the linear encodings of the inconsistent cycles define the DCOP constraints. Soft constraints can then be added to minimize the sum of bound adjustments, as in Equation (5).

4.1 The Weak Repair Problem as a DCOP

Given a non-WC MISTNU $\mathcal{G} = (A, \Sigma, \mathcal{B})$, we build the corresponding DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \mu \rangle$ as follows. The DCOP agents are the MISTNU agents, i.e., $\mathcal{A} = A$. For each contract $p \in \bigcup_{a \in A} P_a$ with $\mathcal{B}(p) = [l_p, u_p]$, we introduce two DCOP variables, l^p and u^p, representing the repaired lower

and upper bounds of p. Since DCOP variables have finite domains, we discretize the interval $[l_p, u_p]$ according to the application's temporal granularity. Thus, $d(l^p) = d(u^p) = \{l_p, \ldots, u_p\}$. This discretization is exact whenever time values are represented as integers in the chosen time unit. Therefore, $\mathcal{X} = \bigcup_{p \in \bigcup_{a \in A} P_a} \{l^p, u^p\}$, $\mathcal{D} = \{d(x) \mid x \in \mathcal{X}\}$.

Next, we determine $\mathcal{F}$ by adding a cost function to the set of linear constraints defined in the Weak Repair problem to use the objective function $F(\sigma)$ introduced in Definition 9. We consider two types of hard constraints: those induced by inconsistent cycles, and those enforcing the consistency of the repaired bounds of each contract, i.e., $l^p \leq u^p$.

Hard constraints are modeled through infinite costs: each hard constraint has cost 0 when it is satisfied and cost $+\infty$ otherwise. For example, the inconsistent cycle of agent a in Figure 2 induces the cost function $f^i_{\text{cycle}} = 0$ if $(l^p - 5) + (l^q - 10) + (l^r - 10) \geq 5$, and $f^i_{\text{cycle}} = +\infty$ otherwise. Similarly, for each contract p, we add a bound-consistency cost function $f^p_{\text{bounds}} = 0$ if $l^p \leq u^p$, and $f^p_{\text{bounds}} = +\infty$ otherwise.

The mapping μ assigns both variables l^p and u^p to the agent that owns contract p. Formally, if $p \in W_a$, then $\mu(l^p) = \mu(u^p) = a$.

At this stage, WRP-DCOP models the satisfaction version of the Weak Repair, using only hard constraints. A DCOP solver searches for an assignment minimizing $F(\sigma)$. In this satisfaction version, a solution with cost 0 satisfies all hard constraints and therefore corresponds to a valid repair. If every assignment has cost $+\infty$, then at least one hard constraint is violated in every possible assignment, and no repair exists.

4.2 Minimizing Contract Reductions

To model a WRP-DCOP that minimizes bound reductions, as in Equation (5), we add soft unary cost functions. For each contract p, the cost associated with increasing the lower bound is $l^p - l_p$, while the cost associated with decreasing the upper bound is $u_p - u^p$. Thus, among all assignments satisfying the hard constraints, the solver selects one that minimizes the total bound adjustment. A finite-cost solution is a valid repair; if all assignments have cost $+\infty$, no repair exists.

Example 2. *In the following, we model the example of the Weak Repair Problem of Figure 1 using the inconsistent cycles of Figure 2 as a WRP-DCOP:*

- $\mathcal{A} = \{a, b, c\}$;
- $\mathcal{X} = \{l^p, u^p, l^q, u^q, l^r, u^r\}$;
- $\mathcal{D} = \{d(x) \mid x \in \mathcal{X}\}$, *where* $d(l^i) = d(u^i) = \{l_i, \ldots, u_i\}$ *for each* $i \in \{p, q, r\}$;
- $\mathcal{F} = \{f^1_{cycle} = 0$ *if* $(l^p - 5) + (l^q - 10) + (l^r - 10) \geq 5$ *else* $+\infty$, $f^2_{cycle} = 0$ *if* $(10 - u^p) + (15 - u^q) + (20 - u^r) \geq 15$ *else* $+\infty$, $f^3_{cycle} = 0$ *if* $(l^p - 5) \geq 1$ *else* $+\infty$, $f^p_{bounds} = 0$ *if* $l^p \leq u^p$ *else* $+\infty$, $f^q_{bounds} = 0$ *if* $l^q \leq u^q$ *else* $+\infty$, $f^r_{bounds} = 0$ *if* $l^r \leq u^r$ *else* $+\infty$, $f^{l^p}_{reduc} : (l^p - l_p)$, $f^{u^p}_{reduc} : (u_p - u^p)$, $f^{l^q}_{reduc} : (l^q - l_q)$, $f^{u^q}_{reduc} : (u_q - u^q)$, $f^{l^r}_{reduc} : (l^r - l_r)$, $f^{u^r}_{reduc} : (u_r - u^r)\}$;
- $\mu = \{l^p \mapsto a, u^p \mapsto a, l^q \mapsto b, u^q \mapsto b, l^r \mapsto a, u^r \mapsto a\}$.

Note that agent c controls no contract in this example and therefore owns no DCOP variable.

5 Experimental Evaluation

Here, we empirically evaluate the proposed centralized and distributed repair approaches. For the centralized approach, we compare our linear encoding with the previous SMT encoding of [Sumic *et al.*, 2024b]. For the distributed approach, we evaluate both the satisfaction version of WRP-DCOP and the optimization version that minimizes the number of bound adjustments. The complexity of our methods depends on two components: the WC checking routine and the repair procedure. WC checking is theoretically exponential [Sumic and Vidal, 2024], and the number of inconsistent cycles returned by the checker may also be exponential. Accordingly, our experimental study has two objectives: (i) assess the practical cost of checking WC on MISTNUs, which requires checking each agent's reified STNU; and (ii) evaluate the scalability of the centralized and distributed repair methods despite the potentially exponential number of inconsistent cycles.

5.1 Environment and Benchmark

We conducted our experiments on a 20-core Intel® Xeon® CPU E5-2660 at 2.6 GHz, with 64 GB RAM, running Ubuntu 18. We implemented our centralized method using the `pySMT` framework [Gario and Micheli, 2015], with Z3 [De Moura and Bjørner, 2008] as the backend solver for linear real arithmetic. For the distributed approach, we used FRODO [Léauté *et al.*, 2009], a Java library providing complete DCOP algorithms.

The benchmark [Sumic *et al.*, 2024b] consists of 350 non-WC but repairable MISTNUs. Each instance involves four agents, and all agent networks in the same instance have the same number of timepoints. The benchmark is organized into seven groups of 50 instances, with network sizes in $(20, 30, 50, 80, 100, 150, 200)$ timepoints per agent network. For each size, the number of owned contracts per agent is fixed at 1-7. Since an activity duration is modeled as a constraint between two timepoints (start and end), a network of size 50 can represent a plan with around 20 activities plus additional deadlines.

5.2 Centralized Method Evaluation

Figure 3 reports the average execution time as a function of the network size. We separate the time required to identify inconsistent cycles, namely WC checking, from the time required to repair them using our linear encoding. The figure also reports the average execution time of the SMT repair algorithm of [Sumic *et al.*, 2024b]. With a time limit of one hour, the SMT encoding repaired instances up to size 30, whereas our centralized method repaired instances up to size 100 and timed out on sizes 150 and 200.

Table 1 provides additional information on solved instances, including the average number of inconsistent cycles and their average arity. These results illustrate the exponential increase in the number of cycles requiring repair.

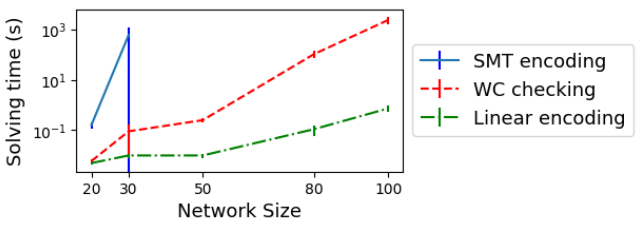

Figure 3: Average execution time, with standard deviation, as a function of network size. The plot separates WC checking, linear repair, and SMT repair. SMT repair exhibits high variance.

Network size	**20**	**30**	**50**	**80**	**100**
Average # cycles	6±2	52±17	75±20	1045±300	7542±1800
Average arity	2±1	3±2	4±2	5±2	7±2

Table 1: Average number and arity of inconsistent cycles.

Metric	**DPOP**	**ADOPT**	**AFB**	**SyncBB**
Instances solved	55/250	138/250	250/250	250/250
Average time (s)	3.5±13.2	12.7±28.8	1.12±2.2	**0.74±1.4**

Table 2: SAT-version results for complete DCOP algorithms.

Overall, the centralized linear encoding outperforms the SMT encoding in efficiency and scalability. Although inconsistent cycles can be exponential in number, repairing one cycle often repairs others involving the same contracts. For instance, in Figure 2, fixing contract p to $[10, 10]$ repairs the inconsistent cycles of agents a and c. Thus, WC checking is the main computational bottleneck and the primary target for scalability improvements.

5.3 Study of the Distributed Approach

We evaluate the distributed approach on the same benchmark instances used for the centralized evaluation, encoded as WRP-DCOP instances as described in Section 4. We consider four complete DCOP algorithms: SyncBB, a synchronous distributed branch-and-bound algorithm [Hirayama and Yokoo, 1997]; DPOP, a synchronous inference-based algorithm over a DFS pseudo-tree [Petcu and Faltings, 2005]; ADOPT, an asynchronous memory-bounded best-first search algorithm [Modi *et al.*, 2005]; and AFB, an asynchronous search algorithm related to SyncBB [Gershman *et al.*, 2006].

We first compare the four algorithms on the satisfaction version of WRP-DCOP, without optimization, to assess the feasibility of repair-based search. With a 5-minute limit per instance, Table 2 reports solved instances and average runtimes, with standard deviations shown after ±, over the 250 instances of sizes 20–100. SyncBB and AFB solve all instances; ADOPT solves fewer; and DPOP solves only 55/250, likely due to high-arity constraints.

Next, we evaluate AFB and SyncBB on the optimization version of WRP-DCOP, minimizing total bound adjustments. With a 1-hour limit per instance, Table 3 shows that AFB solves more instances and is faster than SyncBB, likely due to concurrent constraint checks. However, neither algorithm returns optimal solutions within the time limit for instances of sizes 80 and 100.

Network size	# solved instances		Average execution time (s)		Solution quality (AFB/SyncBB)
	AFB	SyncBB	AFB	SyncBB	
20	50/50	50/50	**55.4**±131.2	87±167.5	-
30	**17**/50	10/50	**941.4**±825.9	1155.6±942.6	75%
50	**12**/50	7/50	**405.3**±372.9	482.3±422.7	72%
80	0/50	0/50	-	-	**82%**
100	0/50	0/50	-	-	**90%**

Table 3: Average number and arity of inconsistent cycles.

For instances where AFB and SyncBB did not return an optimal solution within the time limit, we compared the feasible solutions from the satisfaction experiment with the corresponding optimal costs computed by the centralized method. Solution quality is reported as $100 \cdot C^\star/C$, where $C^\star$ is the optimal repair cost and C is the cost of the feasible solution; higher values are better. As shown in Table 3, both algorithms produced identical quality values, with particularly high-quality feasible repairs for instances of sizes 80 and 100.

Overall, the scalability of the distributed approach is mainly limited by the underlying DCOP algorithms when optimality is required. However, feasible, non-optimal repairs provide a practical trade-off: they can be obtained within a few seconds and achieve reasonably high solution quality. For sizes 80 and 100, feasible distributed repairs still reach 82%–90% of the centralized optimum.

6 Conclusions

This paper addressed the Weak Repair Problem for non-WC MISTNUs through a cycle-based perspective. Starting from WC-checking algorithms that identify inconsistent cycles, we provided a linear encoding of those cycles and used it to derive two repair methods: a centralized linear-programming approach that computes optimal repairs, and a distributed formulation, WRP-DCOP, that preserves privacy while enabling inter-agent coordination.

The empirical results support the practical relevance of this framework. The centralized method is more efficient and scalable than the previous SMT encoding, repairing instances up to size 100 within the time limit (while SMT stops at size 30). In the distributed setting, complete DCOP solvers are effective for finding feasible repairs, and, when optimality is difficult to guarantee on larger instances, they still return solutions with good quality. Overall, these results indicate that repairing through inconsistent cycles is a promising alternative to blind repair.

The main limitation highlighted by our study is not repair itself, but WC checking, which remains the main computational bottleneck. Future work will therefore focus on: (i) tailored WRP-DCOP algorithms and better ordering heuristics; (ii) incomplete distributed methods that trade optimality for scalability; and (iii) extending the same cycle-based repair principle to Dynamic Controllability via iterative detect-and-repair schemes. Improving cycle-detection methods to return richer sets of relevant cycles appears particularly promising for scaling both centralized and distributed repair.

References

[Barrett and Tinelli, 2018] Clark Barrett and Cesare Tinelli. *Satisfiability modulo theories*. Springer, 2018.

[Bhargava and Williams, 2019] Nikhil Bhargava and Brian C. Williams. Multiagent disjunctive temporal networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, Canada*, 2019.

[Botti *et al.*, 1995] Vicente Juan Botti, Federico Barber, Alfons Crespo, Eva Onaindia, Ana García-Fornes, Ismael Ripoll, Daniela Gallardo, and Luis Hernández. A temporal blackboard for a multi-agent environment. *Data & Knowledge Engineering*, 15:189–211, 1995.

[Casanova *et al.*, 2016] 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*. IOS Press, 2016. DOI: 10.3233/978-1-61499-672-9-930.

[De Moura and Bjørner, 2008] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008.

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

[Gario and Micheli, 2015] Marco Gario and Andrea Micheli. pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop*, 2015.

[Gershman *et al.*, 2006] Amir Gershman, Amnon Meisels, and Roie Zivan. Asynchronous forward-bounding for distributed constraints optimization. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*, page 103–107, NLD, 2006. IOS Press.

[Hirayama and Yokoo, 1997] Katsutoshi Hirayama and Makoto Yokoo. Distributed partial constraint satisfaction problem. In Gert Smolka, editor, *Principles and Practice of Constraint Programming - CP97, Third International Conference, Linz, Austria, October 29 - November 1, 1997, Proceedings*, volume 1330 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 1997. DOI: 10.1007/BFB0017442.

[Hunsberger and Posenato, 2022] Luke Hunsberger and Roberto Posenato. Speeding Up the RUL$^-$ Dynamic-Controllability-Checking Algorithm for Simple Temporal Networks with Uncertainty. In *36th AAAI Conf. on Artificial Intelligence 2022*, pages 9776–9785. AAAI Press, 2022. DOI: 10.1609/aaai.v36i9.21213.

[Léauté *et al.*, 2009] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek. FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*, pages 160–164, Pasadena, California, USA, July 13 2009. https://frodo-ai.tech.

[Lubas and Eder, 2023] Josef Lubas and Johann Eder. A time-aware model for legal smart contracts. In *Enterprise, Business-Process and Information Systems Modeling - 24th International Conference BPMDS 2023*, Lecture Notes in Business Information Processing. Springer, 2023. DOI: 10.1007/978-3-031-34241-7_9.

[Maris *et al.*, 2025] Frédéric Maris, Aïdin Sumic, Thierry Vidal, and Bruno Zanuttini. Study of the computational complexity of the repair problem on simple temporal networks with uncertainty. In *19èmes Journées d'Intelligence Artificielle Fondamentale et 20èmes Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes, JIAF-JFPDA 2025, Dijon, France, July 2-4, 2025*, pages 101–109, 2025.

[Modi *et al.*, 2005] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, 2005.

[Petcu and Faltings, 2005] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 266–271. Professional Book Center, 2005.

[Sumic and Vidal, 2024] Ajdin Sumic and Thierry Vidal. A more efficient and informed algorithm to check weak controllability of simple temporal networks with uncertainty. In *31st International Symposium on Temporal Representation and Reasoning, TIME 2024, Montpellier, France, October 28-30, 2024*, LIPIcs, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. DOI: 10.4230/LIPICS.TIME.2024.8.

[Sumic *et al.*, 2024a] Ajdin Sumic, Alessandro Cimatti, Andrea Micheli, and Thierry Vidal. SMT-based repair of disjunctive temporal networks with uncertainty: Strong and weak controllability. In *21st International Conference, CPAIOR 2024*, Lecture Notes in Computer Science. Springer, 2024.

[Sumic *et al.*, 2024b] Ajdin Sumic, Thierry Vidal, Andrea Micheli, and Alessandro Cimatti. Introducing interdependent simple temporal networks with uncertainty for multiagent temporal planning. In *31st International Symposium on Temporal Representation and Reasoning*, 2024.

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

[Yokoo *et al.*, 1998] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algo-

rithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.

[Zhang and Williams, 2021] Yuening Zhang and Brian C. Williams. Privacy-preserving algorithm for decoupling of multi-agent plans with uncertainty. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling, ICAPS 2021, Guangzhou, China (virtual), August 2-13, 2021*. AAAI Press, 2021.