Solving dynamic controllability problem of multi-agent plans with uncertainty using mixed integer linear programming.

Guillaume Casanova, Cédric Pralet, Charles Lesire, Thierry Vidal

Solving Dynamic Controllability Problem
of Multi-Agent Plans with Uncertainty
Using Mixed Integer Linear Programming

Guillaume Casanova [1] and **Cédric Pralet** [1] and **Charles Lesire** [1] and **Thierry Vidal** [2]

Abstract.

Executing multi-agent missions requires managing the uncertainty about uncontrollable events. When communications are intermittent, it additionally requires for each agent to act only based on its local view of the problem, that is independently of events which are controlled or observed by the other agents. In this paper, we propose a new framework for dealing with such contexts, with a focus on mission plans involving temporal constraints. This framework, called Multi-agent Simple Temporal Network with Uncertainty (MaSTNU), is a combination between Multi-agent Simple Temporal Network (MaSTN) and Simple Temporal Network with Uncertainty (STNU). We define the dynamic controllability property for MaSTNU, and a method for computing offline valid execution strategies which are then dispatched between agents. This method is based on a mixed-integer linear programming formulation and can also be used to optimize criteria such as the temporal flexibility of multi-agent plans.

1 Introduction

In robotic applications such as the autonomous exploration of large and hazardous areas, better performances can be obtained by using multiple robots. This can indeed lead to a faster achievement of the mission due to parallel realizations of tasks, and bring redundancy for continuing the mission in case of robot failures. One difficulty to overcome in this context is that the tasks allocated to robots must be coordinated, since there may exist precedence or synchronization constraints between tasks, or more generally constraints on the minimum/maximum temporal distances between tasks. To handle these multi-agent temporal constraints, *Multi-agent Simple Temporal Networks* (MaSTNs [1]) were recently introduced, with techniques for computing, in a distributed way, allowed distances between time-points involved in plans [1], or earliest/latest occurrence times of time-points [2].

However, one issue when using MaSTN for robotic missions is that MaSTN are not designed for obtaining decision strategies which are robust to the uncertainty about the occurrence time of uncontrollable time-points. For instance, they are not adapted to obtain plans which are feasible whatever the exact duration of tasks turn out to be. Along this line, they are not as expressive as the framework of Simple Temporal Network with Uncertainty (STNU [9]),

which makes an explicit distinction between *executable* time-points, which can be directly controlled, and *contingent* time-points, which cannot. In STNU, robust execution strategies describe a way to set the occurrence time of executable time-points depending on time-point occurences observed so far, and these strategies are built under the strong assumption that the realization of every time-point in the temporal network is instantaneously observed. Such an assumption is often violated for multi-robot systems, since each event might be observable only from a restricted set of geographical positions.

This is why we propose a new framework for managing temporal constraints over multi-agent systems. This framework, called *Multi-agent Simple Temporal Network with Uncertainty* (MaSTNU), can be seen as an attempt to combine MaSTN and STNU. It is equipped with algorithms to compute robust execution strategies which respect every temporal constraint of the multi-agent plan despite the uncertainty about the occurrence time of contingent time-points, and which are applicable even in constrained environments featuring intermittent communications between agents. Such distributed execution strategies are obtained using a centralized offline procedure based on a Mixed Integer Programming (MIP) formulation of what we call the *multi-agent dynamic controllability problem*. This procedure is run at the mission center before triggering the coordinated deployment of the agents on the field.

The paper is organized as follows. Sect. 2 introduces some background on STN, MaSTN, STNU, and dynamic controllability checking. Sect. 3 presents the MaSTNU framework. Sect. 4 details our MIP approach for dealing with multi-agent dynamic controllability. Sect. 5 provides experimental results and discusses several ways to optimize execution strategies.

2 Background

2.1 Simple Temporal Network

A standard framework for reasoning about temporal constraints is the framework of *Simple Temporal Problems* (STPs [4]). Basically, an STP is a pair $S = (V, E)$ defined by a set $V = \{v_1, \ldots, v_n\}$ of time-points representing event occurrence times, and a set E of temporal constraints between these time-points. Each constraint $e \in E$ takes the form $v_j - v_i \in [L_{ij}, U_{ij}]$, where $L_{ij} \in \mathbb{R} \cup \{-\infty\}$ and $U_{ij} \in \mathbb{R} \cup \{+\infty\}$ respectively specify a minimum and a maximum temporal distance between v_i and v_j. A specific time-point v_0 called the *reference-point* is usually added to V for representing a reference temporal position, and unary temporal constraints such as $v_i \in [L_{0i}, U_{0i}]$ can then be easily expressed as distance constraints with regards to this reference-point (constraints $v_i - v_0 \in [L_{0i}, U_{0i}]$).

[1] Onera – The French Aerospace Lab, F-31055, Toulouse, France, email: name.surname@onera.fr
[2] Ecole Nationale dIngenieurs de Tarbes, Tarbes, France, email: name.surname@enit.fr

STPs have a natural graphical representation called *Simple Temporal Networks* (STNs), which contain one vertex per time-point in V and one edge $v_i \rightarrow v_j$ labeled by $[L_{ij}, U_{ij}]$ per temporal constraint in E. STNs are appealing in practice to deal with temporal aspects because several problems that can be formulated on STNs are solvable in polytime [4], such as determining whether there exists an assignment of time-points satisfying all temporal constraints.

STNs were extended to deal with multi-agent problems on one hand and with uncertain temporal durations on the other hand. Thereafter, we give some background on these two distinct extensions.

2.2 Multi-agent Simple Temporal Network

STNs were extended to a multi-agent context, where time-points are not controlled by a single agent but are instead partitioned among a set of agents $\mathcal{A}$. This extension is called MaSTN for *Multiagent Simple Temporal Network* [1]. Formally, an MaSTN is defined by:

- a set of *local STNs* (one per agent $a \in \mathcal{A}$); the *local* STN associated with agent a, denoted by S_L^a, is defined by V^a, the set of *local* time-points owned by a, and E_L^a, the set of *local* edges which connect only time-points in V^a;
- a set of *external* edges E_X, each of which constrains the temporal distance between two time-points belonging to distinct agents; apart from its local edges in E_L^a, each agent a is aware of the subset of external constraints which hold on one of its local vertices.

Fig. 1 gives an example of an MaSTN involving three agents A, B, C. Agent A (resp. B and C) owns variables v_1^A to v_6^A (resp. v_1^B to v_8^B and v_1^C to v_8^C). Edge (v_1^A, v_2^A) is an example of a local edge for agent A. Edge (v_3^A, v_3^B) is an example of an external edge in E_X. It enforces some synchronization between agents A and B.

MaSTN algorithms were defined to compute, in a distributed way, possible temporal distance between pairs of time-points (distributed partial path-consistency algorithms [1]), as well as earliest/latest dates associated with time-points (distributed arc-consistency algorithms [2]).
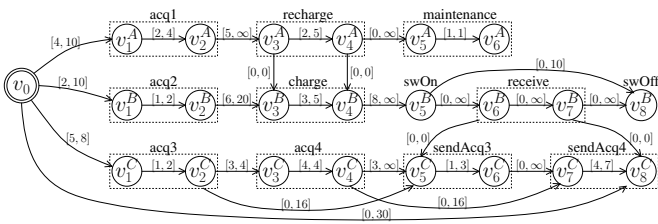

Figure 1. Example of a Multi-agent STN involving 3 agents A, B, C (one line per agent)

2.3 Simple Temporal Network with Uncertainty

In another direction, STNs were extended to *Simple Temporal Networks with Uncertainty* (STNUs [9]) in order to represent *uncertain* durations, that is durations whose value is fixed by an external process rather than by the planning agent itself.

Formally, an STNU is a triple (V, E, C), where V is a set of time-points, E is a set of *requirement* links, and C is a set of *contingent* links. Each requirement link is defined as in standard STN. Each contingent link is defined by a pair of time-points (v_i, v_j) and by a temporal interval $[L_{ij}, U_{ij}]$ with $0 < L_{ij} < U_{ij} < \infty$. The duration

of such a contingent link, that is distance $v_j - v_i$, is known to be between L_{ij} and U_{ij}, but its precise value is not controlled. In this case, v_i and v_j are respectively called the *activation* time-point and the *contingent* time-point. Last, a time-point cannot be the contingent time-point of two distinct contingent links. Any time-point which is not a contingent time-point for some contingent link is called an *executable* time-point. In the following, we denote by V_E the set of executable time-points and by V_C the set of contingent time-points.

Fig. 2 gives an example of an STNU involving six time-points plus the reference time-point v_0. Requirement links such as $v_2^B - v_1^B \in [4, 6]$ are depicted using continuous lines, while contingent links such as $v_1^B - v_1^A \in [1, 4]$ are depicted using dashed lines. In this STNU, the set of executable time-points is $V_E = \{v_1^A, v_2^A, v_2^B, v_3^B\}$ and the set of contingent time-points is $V_C = \{v_1^B, v_3^A\}$.

The fundamental problem associated with an STNU is to determine whether it is *dynamically controllable*, which informally means that there exists a way to dynamically assign values to executable time-points depending on observations collected, so that all requirement links are satisfied whatever the precise values of contingent links turn out to be at execution time.

More formally, dynamic controllability over STNU can be defined as follows. First, a *projection* of an STNU is an STN obtained by replacing each contingent link $v_j - v_i \in [L_{ij}, U_{ij}]$ by a deterministic link $v_j - v_i \in [d, d]$ with $d \in [L_{ij}, U_{ij}]$. A *schedule* is an assignment of values to all time-points. An execution strategy R can then be defined as a mapping from projections to schedules. An execution strategy R is said to be *valid* when for every projection p, schedule $R(p)$ satisfies all requirement links. An execution strategy R is said to be *dynamic* iff for every executable time-point v and every projections p_1, p_2 of the STNU, if the assignment of all time-points scheduled before v are the same in $R(p_1)$ and $R(p_2)$, then the values assigned to v in $R(p_1)$ and $R(p_2)$ are the same. In other words, the execution time of v can only depend on the information gathered before executing v. Last, an STNU is Dynamically Controllable (DC) iff there exists an execution strategy with is both valid and dynamic.

The STNU provided in Fig. 2 is dynamically controllable, and a valid dynamic execution strategy can be: (a) execute v_1^A at time 0, (b) wait for v_1^B to happen, (c) execute v_2^B at time $v_1^B + 4$, (d) execute v_2^A at time $v_2^B + 6$, (e) execute v_3^B at time $v_2^B + 6$, (f) wait for v_3^A to happen.
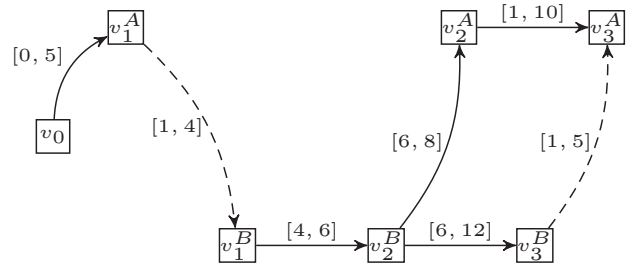

Figure 2. Example of dynamically controllable STNU

2.4 Checking Dynamic Controllability using MIP

Several algorithms do exist for checking dynamic controllability [9, 7, 5]. One of them consists in using graph-based algorithms for com-

puting mandatory *wait* constraints on requirement links. A wait constraint (v_k, w_{ijk}) on a requirement link $(v_i, v_j) \in E$ means that v_j can only be executed either after v_k is executed, or after w_{ijk} time units since the execution of v_i.

In a completely different direction, DC checking on STNUs can also be formulated as a Mixed-Integer linear Program (MIP) [3]. One advantage of such a MIP formulation is that it can be adapted for answering more general queries, such as minimally updating time bounds on contingent links such that a non-DC STNU becomes DC.

Fig. 3 gives the disjunctive linear model introduced in [3], from which a MIP model can be obtained using some linearization steps. Roughly speaking, the model contains two continuous decision variables l_{ij} and u_{ij} for each pair of time-points (v_i, v_j). Variables l_{ij} and u_{ij} respectively represent the lower and upper bounds imposed on the distance $v_j - v_i$ between time-points v_i and v_j. The model also contains a set of continuous wait variables w_{ijk} (one variable per triple of time-points (v_i, v_j, v_k) such that v_k is a contingent time-point). These variables have the same meaning as seen previously. Discrete decision variables are present in the MIP model after the linearization process. If a solution is found to the problem, then the STNU is DC. See [3] for details concerning the correctness of the modeling and the linearization process.

An optimization function f_{opt} can easily be added to the model, for instance to maximize the flexibility of solutions by using $f_{opt} = \sum_{i<j} (u_{ij} - l_{ij})$.

$$\forall (v_i, v_j) \in E, \ L_{ij} \leq l_{ij} \leq u_{ij} \leq U_{ij} \tag{1}$$

$$\forall (v_i, v_j) \in C, \ (l_{ij} = L_{ij}) \wedge (u_{ij} = U_{ij}) \tag{2}$$

$$\forall v_i, v_j, v_k \in V, \begin{cases} l_{ik} \leq u_{ij} + l_{jk} \leq u_{ik} \\ l_{ik} \leq l_{ij} + u_{jk} \leq u_{ik} \\ u_{ik} \leq u_{ij} + u_{jk} \\ l_{ij} + l_{jk} \leq l_{ik} \end{cases} \tag{3}$$

$$\forall (v_i, v_k) \in C, \forall v_j \in V_E, \ (l_{jk} < 0) \vee \begin{pmatrix} u_{ij} \leq l_{ik} - l_{jk} \\ l_{ij} \geq u_{ik} - u_{jk} \end{pmatrix} \tag{4}$$

$$\forall (v_i, v_k) \in C, \forall v_j \in V_E, \ u_{ik} - u_{jk} \leq w_{ijk} \tag{5}$$

$$\forall (v_i, v_j) \in E, \forall v_k \in V_C, \ \min(l_{ik}, w_{ijk}) \leq l_{ij} \tag{6}$$

$$\forall (v_i, v_k), (v_m, v_j) \in C^2, \\ (w_{ijk} < 0) \vee (w_{ijk} - l_{mj} \leq w_{imk}) \tag{7}$$

$$\forall (v_i, v_k) \in C, \forall v_m, v_j \in V, \ w_{ijk} - u_{mj} \leq w_{imk} \tag{8}$$

Figure 3. Disjunctive linear model for encoding DC on STNU [3]

3 Multi-agent Simple Temporal Network with Uncertainty (MaSTNU)

3.1 Framework Definition

As explained in the introduction, STNU cannot be directly reused in a multi-agent setting, where each agent only controls a subset of the executable time-points and only observes the occurrence of a subset of the contingent time-points. This is why we introduce Multi-agent STNU (MaSTNU).

Formally, an MaSTNU is a quadruplet $(\mathcal{A}, V, E, C)$, with $\mathcal{A}$ a set of agents and (V, E, C) an STNU (V denotes the set of executable and contingent time-points, E the set of requirement links, and C the set of contingent links). Additionally, as in MaSTN, time-points in V are partitioned among $\mathcal{A}$, that is for every time-point $v \in V$ there exists a unique agent $a \in \mathcal{A}$ which *owns* v, denoted by $owner(v) = a$. Semantically speaking, if v is an executable time-point, then the owner of v is the agent which controls the execution of the event associated with v. If v is a contingent time-point, the owner of v is the unique agent which is assumed to instantaneously observe the realization of v. Time-points owned by other agents are not supposed to be directly observed, however information about their realization can be obtained thanks to external contingent links. Reference-point v_0 represents a clock synchronized between agent and is considered to be simultaneously owned by all agents.

In the following, for each agent $a \in \mathcal{A}$, V^a denotes the set of time-points owned by a, called the *local time-points* of a. We denote by E_L^a (resp. C_L^a) the set of *local* requirement links (resp. contingent links), which hold only on time-points owned by a. Analogously to MaSTN, we also define E_X (resp. C_X) as the set of *external* requirement links (resp. contingent links), which connect two time-points owned by different agents.

Fig. 4 gives an example of an MaSTNU involving two agents A and B, which respectively own time-points $V^A = \{v_1^A, v_2^A, v_3^A\}$ and $V^B = \{v_1^B, v_2^B, v_3^B\}$. The link from v_1^A to v_1^B is an external contingent link, the link from v_2^B to v_2^A is an external requirement link, the link from v_1^B to v_2^B is a local requirement link, and there is no local contingent link. Semantically speaking, external contingent links model observations received by an agent, the source of these observations being owned by other agents. For instance, the source of a contingent link might be the start of a data transmission process triggered by one agent, and the target of this link might be the end of this data transmission process, observed by the receiving agent. As in MaSTN, external requirement links correspond to synchronization constraints between agents. For instance, they can serve to express that there must not be more than 10 time units between successive surveillances of a given area by two distinct agents.
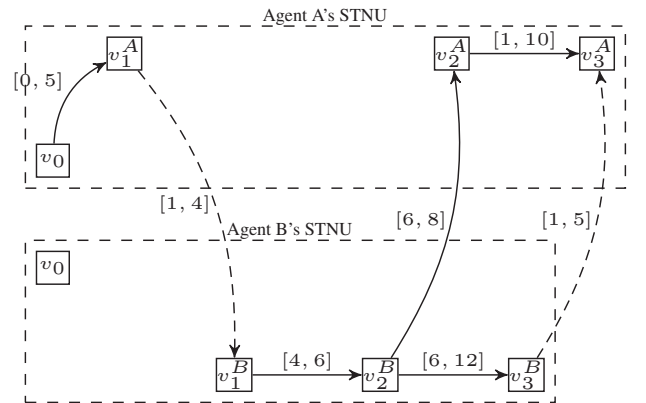

Figure 4. Example of an MaSTNU

3.2 Dynamic Controllability Revisited

The multi-agent nature of MaSTNU requires an adaptation of the dynamic controllability property. Indeed, given an MaSTNU $(\mathcal{A}, V, E, C)$, computing a valid dynamic execution strategy for STNU (V, E, C) does not necessarily give an applicable multi-agent strategy. As an illustration, consider the MaSTNU provided in Fig. 4. The STNU associated with it is the STNU previously shown in Fig. 2. By considering the execution strategy seen for this STNU and by partitioning it between agents, we obtain the following strategy:

- for agent A: (a) execute v_1^A at time 0, (b) execute v_2^A at time $v_2^B + 6$, (c) wait for v_3^A to happen;
- for agent B: (a) wait for v_1^B to happen, (b) execute v_2^B at time $v_1^B + 4$, (c) execute v_3^B at time $v_2^B + 6$.

The issue with such a strategy is that agent A has no guarantee to be able to execute it, because it might not observe external time-point v_2^B owned by agent B.

This is why we introduce a new definition of dynamic controllability which is adapted to MaSTNU. Let $(\mathcal{A}, V, E, C)$ be an MaSTNU and let R be an execution strategy for the associated STNU (V, E, C). Execution strategy R is said to be *distributed* iff for every projections p, p' (that is for every two possible assignments of the duration of contingent links) and for every agent $a \in \mathcal{A}$, if schedule $R(p)$ and schedule $R(p')$ assign the same value to all contingent time-points owned by a, then they also assign the same value to all executable time-points owned by a. In other words, each agent only acts based on its own immediate observations, which means that the execution strategy is robust to the missing observations of external time-points.

An MaSTNU $(\mathcal{A}, V, E, C)$ is then said to be *dynamically controllable* iff STNU (V, E, C) admits an execution strategy which is *valid* (it induces schedules which satisfy all requirement links), *dynamic* (decisions are made only based on past information), and *distributed* (previous definition).

For the MaSTNU given in Fig. 4, an example of a valid, dynamic and distributed execution strategy is:

- for agent A: (a) execute v_1^A at time 4, (b) execute v_2^A at time 19, (c) wait for v_3^A to happen;
- for agent B: (a) wait for v_1^B to happen, (b) execute v_2^B at time $v_1^B + 6$ if $v_1^B \leq 7$ and at time $v_1^B + 4$ otherwise, (c) execute v_3^B at time $v_2^B + 8$.

In the following, we introduce techniques for checking DC for MaSTNU and for automatically computing distributed strategies.

4 Dynamic Controllability Check and Computation of Execution Strategies

To check DC for an MaSTNU $(\mathcal{A}, V, E, C)$, we first check DC for STNU (V, E, C). If this STNU is not DC, then the MaSTNU is not DC either, because acceptable execution strategies for MaSTNU are more restricted than acceptable execution strategies for STNU. Otherwise, if STNU (V, E, C) is DC, we perform additional operations to determine whether the original MaSTNU is DC.

4.1 From one MaSTNU to a set of local STNUs

The key idea in our method is to transform the original MaSTNU S into a *distributed* MaSTNU, which contains no external link between agents, and then to partition this distributed MaSTNU into a set of local STNUs $\{S^a \mid a \in \mathcal{A}\}$. Fig. 5 shows an example of such a process. The reason why we consider distributed MaSTNUs as the transformation target is that if each agent $a \in \mathcal{A}$ uses a valid dynamic execution strategy R^a for its own local STNU S^a, then the global strategy obtained by joining strategies R^a is valid and dynamic, and it is also distributed because we are sure that in R^a, each agent acts only based on the observations it is supposed to get at execution (no possible occurrence of external time-points in the execution strategy thanks to the partitioning). In other words, the set of local execution strategies $\{R^a \mid a \in \mathcal{A}\}$ allows to dynamically control the MaSTNU.

Globally, to transform the original MaSTNU into a set of local STNUs, we need to perform two kinds of operations:

1. to replace external requirement links of the original MaSTNU by requirement links which are local to agents, as done in Fig. 5 for external constraint $v_2^A - v_2^B \in [6, 8]$ which will necessarily be satisfied thanks to the two internal requirements $v_2^A - v_0 \in [19, 19]$ and $v_2^B - v_0 \in [11, 13]$ which are present in the partitioned MaSTNU; the introduced local requirement links can be stronger than in the initial MaSTNU, and they are implicitly used to coordinate agent actions;
2. to remove external contingent links and to replace them by local contingent links, as done in Fig. 5 for (v_1^A, v_1^B) which is replaced by (v_0, v_1^B); more generally, the external source v of a contingent link (v, w) must be replaced by a local source u contained in the agent which owns w.

The way these two operations are realized is presented in the two following sections. Compared to DC reasoning on STNU, it is worth mentioning that the transformation of the original MaSTNU into several local STNUs is a *combinatorial decision problem*, because for instance there is not necessarily a unique way of distributing/sharing the satisfaction of external requirement links among agents, or a unique way of reassigning the contingency source of a contingent time-point. The associated decision problem is formalized using a MIP model, which allows us to reuse elements from the existing MIP model given in Sect. 2.4 for standard STNUs. In the MIP model built, we capture several constraints guaranteeing the satisfaction of the external requirement links of the original MaSTNU, and several constraints guaranteeing that the local contingency assumptions made in the distributed MaSTNU are not restrictive with regards to the set of possible scenarios covered by the external contingent links of the original MaSTNU. By adding a linear optimization function, MIP solvers can then be used to find an optimal distribution of temporal constraints such that all local STNUs are DC.

In the following, as in the MIP model of DC for STNU, we use, for every $i < j$, variables l_{ij} and u_{ij} to represent the lower and upper bounds imposed on the distance $v_j - v_i$ between v_i and v_j. Moreover, for $i < j$, we also use u_{ji} as a substitute for $-l_{ij}$.

4.2 Internalization of external requirement links

Let us consider an external requirement link $e = (v_i, v_j)$, with $owner(v_i) = a$, $owner(v_j) = b$, $a \neq b$. Initially, e is labeled by $[L_{ij}, U_{ij}]$. The main issue is that a and b might not have enough information during execution to ensure that e is respected. For example, if b waits to observes v_i before executing v_j then it might fail at respecting e if the delay for observing v_i is greater than U_{ij}. Similarly, a has no information about when it should be executing v_i in a way such that b can execute v_j and respect e.

To make sure that the upper bound of e is respected during execution without using any communication between a and b, it suffices to
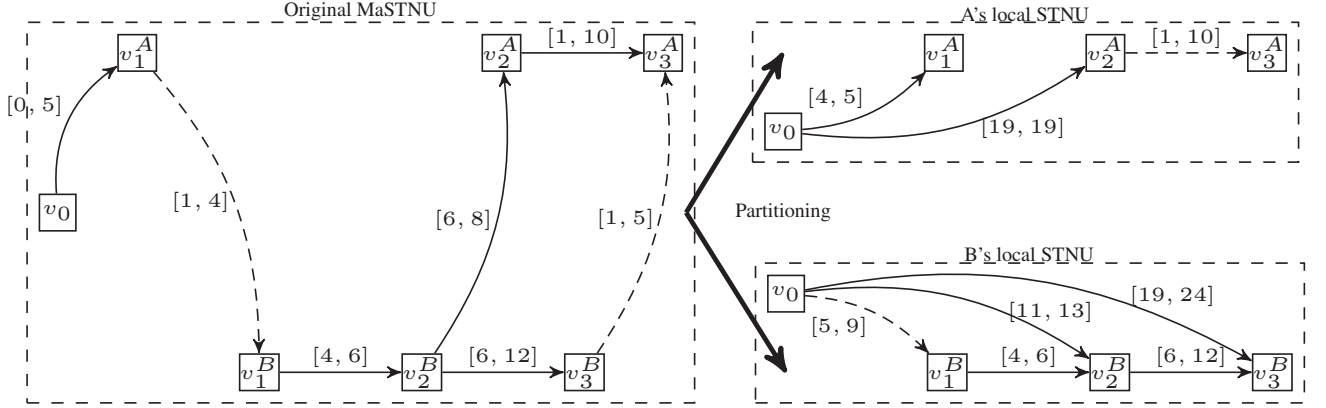
Figure 5. Original MaSTNU and its partitioning

find a path $p = [p_1, ..., p_k]$ composed of time-points, such that:

1. each link (p_i, p_{i+1}) involved in p is either an internal link $(owner(p_i) = owner(p_{i+1}))$, or an external contingent link $((p_i, p_{i+1}) \in C_X$ or $(p_{i+1}, p_i) \in C_X)$;
2. path p defines a path from v_i to v_j which is shorter than U_{ij}, so that if all constraints between time-points in p are satisfied, then e is also satisfied.

Such paths are called *distributed paths*. A contingent link in a distributed path is necessarily satisfied at execution by definition. An internal requirement link in a distributed path is satisfiable by the agent holding it at execution as long as its local STNU is DC. Therefore, if all local STNUs are DC, then all constraints in path p are satisfiable at execution, and therefore the original external requirement link is satisfiable as well. Similar distributed paths must be found to justify that the lower bound of e is satisfied.

In order to find such paths, it actually suffices to decide on the sequence of contingent links to use for justifying the satisfaction of bound u_{ij}, because successive links which are internal to a single agent can be harmlessly collapsed into a single internal link thanks to the path consistency property over STNU. See Figure 6 for an illustration of a path $p = [v_i, v_{k_1}, v_{l_1}, v_{k_2}, v_{l_2}, \ldots, v_{k_n}, v_{l_n}, v_j]$ covering the satisfaction of requirement link (v_i, v_j). On this example, the satisfaction of requirement link (v_i, v_j) is covered by the satisfaction of some requirement over (v_i, v_{k_1}) and (v_{l_1}, v_j); the satisfaction of (v_{l_1}, v_j) is itself covered by the satisfaction of some requirement over (v_{l_1}, v_{k_2}) and (v_{l_2}, v_j)... and so on until there is no more external link to satisfy. Informally speaking, the path built use a sequence of quadrilaterals, and it introduces some new temporal distance constraints with regards to time-points which are correlated through contingent links. Exploiting these correlations is the only way to be robust to the absence of communication. Also, thanks to the path consistency property again, by imposing that local STNUs must be DC, it suffices to search for distributed paths which cross each agent at most once.

In order to formalize such a process, we define two sets:

- set $\overline{E_X}$ which contains all possible external links which are not contingent links, and therefore might require justification; this set is given by $\overline{E_X} = \{(v_i, v_j) \in V^2 \,|\, (owner(v_i) \neq owner(v_j)) \wedge ((v_i, v_j) \notin C_X)) \wedge ((v_i, v_j) \notin C_X)\}$;
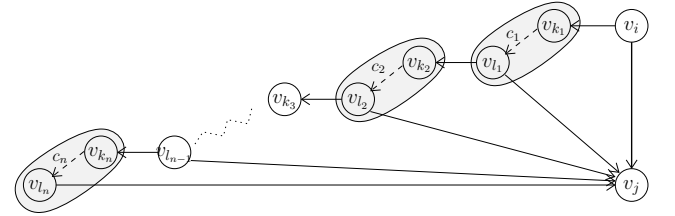

Figure 6. Justification path for an external requirement link between v_i and v_j

- set Q which contains all quadruplets (v_i, v_j, v_k, v_l) such that
  - $(v_i, v_j) \in \overline{E_X}$ is an external requirement link whose satisfaction which might have to be justified when building justification paths;
  - (v_k, v_l) is associated with a contingent link which can be used in the justification for (v_i, v_j), which means that (1) either $(v_k, v_l) = (v_0, v_0)$ (case in which the upper bound u_{ij} over $v_j - v_i$ is justified by a path through the reference time-point), (2) or $(owner(v_k) = owner(v_i)) \wedge ((v_k, v_l) \in C_X \vee (v_l, v_k) \in C_X)$.

Note that the size of Q is at most cubic in the number of time-points, since v_k and v_l are related by a contingent link and because a contingent time-point can only have a unique contingent link pointing to it.

To model the requirement to cover external requirement links by distributed paths, we introduce a MIP modeling which uses the following variables:

- $\forall (v_i, v_j) \in \overline{E_X}, b_{ij} \in \{0, 1\}$ is a boolean decision variable encoding that we need to justify external requirement link $v_j - v_i \leq u_{ij}$;
- $\forall (v_i, v_j, v_k, v_l) \in Q, z_{ijkl} \in \{0, 1\}$ is a boolean decision variable encoding that contingent link between v_k and v_l (either link (v_k, v_l) or link (v_l, v_k)) is used to justify the satisfaction of $v_j - v_i \leq u_{ij}$;
- $\forall (v_i, v_j) \in \overline{E_X}, h_{ij} \in [0, H]$ are decision variables encoding the *height* of the justification of the satisfaction of $v_j - v_i \leq u_{ij}$, with H a constant equal to $\max(|\mathcal{A}| - 2, |C_X|)$; these height variables

are used to avoid cycles, that is to avoid cases in which the satisfaction of the upper bound on an external link e is justified by the upper bound associated with an external link e', and in which the upper bound of e' is justified by the upper bound of e; see Fig. 7 for an illustration of what could happen without preventing cycles in justifications; it also helps bounding the search process since it suffices to consider distributed paths which cross each agent at most once and each contingent link at most once, which explains the value chosen for the upper bound of h_{ij}.
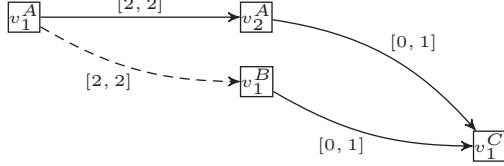

Figure 7. Cycles in justifications without the use of heights variables: the satisfaction of external requirement $v_1^C - v_2^A \leq 1$ can be justified by the satisfaction of external requirement $v_1^C - v_1^B \leq 1$, and reciprocally

We impose several linear constraints for representing the satisfaction of the external requirements by distributed paths. First, the lower and upper bounds associated with contingent links of the original MaSTNU cannot be shrinked:

$$\forall(v_i, v_j) \in C_X, (l_{ij} \leq L_{ij}) \wedge (u_{ij} \geq U_{ij}) \qquad (9)$$

Next, every external requirement link in the initial MaSTNU must be justified:

$$\forall(v_i, v_j) \in E_X \text{ s.t. } U_{ij} \neq +\infty, \, b_{ij} = 1 \qquad (10)$$

$$\forall(v_i, v_j) \in E_X \text{ s.t. } L_{ij} \neq -\infty, \, b_{ji} = 1 \qquad (11)$$

If an external requirement link (original or intermediate) must be justified, then there exists a unique contingent link justifying it:

$$\forall(i, j) \in \overline{E_X}, \, b_{ij} = \sum_{(v_i, v_j, v_k, v_l) \in Q} z_{ijkl} \qquad (12)$$

An external requirement link is justified if there exists a shorter distributed path:

$$\forall(i, j, k, l) \in Q, \, u_{ij} \geq u_{ik} + u_{kl} + u_{lj} + (z_{ijkl} - 1)M \qquad (13)$$

In the previous equation, M is a large constant equal to $L_{ij} - U_{ik} - U_{kl} - U_{lj}$, so that the constraint is always satisfied when $z_{ijkl} = 0$.

Then, every external requirement link used in a justification must also be justified (again, see Fig. 6 for an illustration):

$$\forall(v_i, v_j, v_k, v_l) \in Q \text{ s.t. } (v_l, v_j) \in \overline{E_X}, \, z_{ijkl} \leq b_{lj} \qquad (14)$$

Finally, we are preventing cycles in justifications thanks to the following set of constraints (in the following equation, H is the maximum value of h_{ij} variables):

$$\forall(v_i, v_j, v_k, v_l) \in Q \text{ s.t. } (v_l, v_j) \in \overline{E_X}$$
$$h_{ij} + (1 - z_{ijkl})(H + 1) \geq h_{lj} + 1 \qquad (15)$$

Fig. 8 shows a representation of the MaSTNU obtained after the internalization process of external requirement links. In this example, A and B tighten internal constraints (v_0, v_1^A), (v_0, v_2^A), (v_0, v_2^B) and (v_0, v_3^B) so that external requirement link (v_2^B, v_2^A) is satisfied at execution. Contrarily to the mono-agent STNU solution seen in Fig. 2, v_2^A has no temporal flexibility anymore.
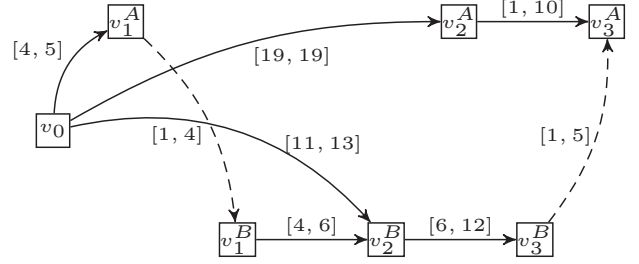

Figure 8. MaSTNU after internalization of external requirement links

4.3 Internalization of external contingent links

External contingent links also have to be internalized, otherwise external time-points might appear in execution strategies, which would invalidate the distributivity of these strategies. Globally, the idea in the internalization of links in C_X is that every potential situation which may be encountered owing to the original MaSTNU must be covered by scenarios considered at the level of local STNUs.

To illustrate the transformation proposed, let us consider an external contingent link $c = (v_i, v_j)$, with $owner(v_i) = a, owner(v_j) = b, a \neq b$. c is labeled by $[L_{ij}, U_{ij}]$, with $L_{ij} > 0$. Any execution strategy directly using the fact that "v_j occurs necessary between L_{ij} and U_{ij} units of time after v_i" cannot be sound as b does not directly observe v_i. This is why we need to explicitly erase c from the MaSTNU representation while keeping the uncontrollable status of v_j. The only solution to do this is to replace link (v_i, v_j) by an internal contingent link (v_k, v_j) in the set of local contingent constraints of agent b. In this case, we say that we use substitution triangle (v_i, v_j, v_k). In the following, we define the set of candidate substitution triangles by $T = \{(v_i, v_j, v_k) \,|\, (v_i, v_j) \in C_X, v_k \in V^{owner(v_j)} \setminus \{v_j\}\}$. For every external contingent link (v_i, v_j), as there is a freedom in the local time-point v_k chosen for activating v_j, we add in the MIP model the following set of decision variables:

- $\forall(v_i, v_j, v_k) \in T, c_{kj} \in \{0, 1\}$ is a boolean decision variable encoding that we substitute external contingent link (v_i, v_j) by a new internal contingent link (v_k, v_j).

Several constraints are imposed over these variables. First, every external contingent link must be substituted by exactly one internal contingent link:

$$\forall(v_i, v_j) \in C_X, \sum_{v_k \,|\, (v_i, v_j, v_k) \in Q} c_{kj} = 1 \qquad (16)$$

If an external contingent link (v_i, v_j) is substituted by an internal contingent link (v_k, v_j), then the bounds specified by (v_i, v_j) must not be less restrictive than the bounds given by path $v_i \to v_k \to v_j$:

$$\forall(v_i, v_j, v_k) \in T, \begin{cases} u_{kj} \geq u_{ki} + u_{ij} + (c_{kj} - 1)M' \\ 0 < l_{kj} \leq l_{ki} + l_{ij} + (1 - c_{kj})M' \end{cases} \qquad (17)$$

with M' a large constant.

If external contingent link (v_i, v_j) is substituted by internal contingent link (v_k, v_j), then the associated requirement link over (v_i, v_k) must be justified:

$$\forall(v_i, v_j, v_k) \in T, \begin{cases} (v_i, v_k) \in \overline{E_X} : c_{kj} \leq b_{ik} \\ (v_k, v_i) \in \overline{E_X} : c_{kj} \leq b_{ki} \end{cases} \qquad (18)$$

Fig. 9 shows the distributed MaSTN obtained after internalizing both external requirement links and external contingent links. Compared to the previous example, (v_0, v_3^B) had to be constrained further. Moreover, (v_0, v_1^B) and (v_2^A, v_3^A) are now contingent links.
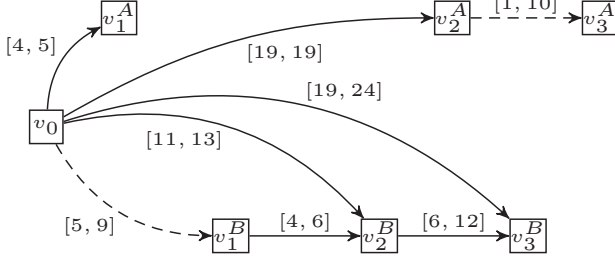

Figure 9. MaSTNU after internalization of external requirement and contingent links

4.4 Dynamic Controllability of Local STNUs

Finally, we have to express that the local STNU associated which each agent must be dynamically controllable. This is expresses by adapting the model provided in Section 2.4. An adaptation is required because due to the choice in the internalization of external contingent links, the set of local contingent link is not fixed. This implies for instance that the constraint given in Eq. 8 must be replaced by:

$$\forall v_k \in C_X, \forall v_i \in V^{owner(v_k)} \setminus \{v_k\}, \forall v_m, v_j \in V,$$
$$w_{ijk} - u_{mj} \le w_{imk} + (1 - c_{ik})M'' \quad (19)$$

with M'' a large constant. Similar transformations must be applied for Eq. 4, 5, and 7.

4.5 Distributing local STNUs

If a solution to the global MIP problem exists, this solution describes a distributed MaSTNU. The latter can be partitioned into a set of N local STNUs, one for each agent, while ignoring external constraints. Local STNUs can be dispatched between agents, and the mission can start.

4.6 Discussion

Completeness The techniques defined for checking DC are sound but not complete, essentially because of the internationalization of external contingent links, which can make lose some information on the correlation between time-points. More precisely, by transforming an MaSTNU into a set of local STNUs, we do not represent some correlations between contingent time-points. For instance, consider an MaSTNU involving one contingent time-point x belonging to agent A, two contingent time-points y, z belonging to agent B, and two contingent links $y - x \in [2, 3]$ and $z - x \in [2, 3]$. In this case, the temporal constraint $z - y \in [-1, 1]$ necessarily holds (y and x are correlated). With our approach, we cannot represent it by a contingent link between y and z (negative lower bound), and we cannot add a new time-point t in B pointing to both y and z since t would have to be observable by B. We believe that all these points are more STNU related issues (representing non-causal uncertainty).

MIP versus propagation techniques for STNU As mentioned previously, searching for robust execution strategies is a combinatorial task and we cannot directly reuse polynomial DC checking techniques available in the literature [9, 7, 5]. Such a combinatorial aspect is also present in [3] for building an STNU which is DC from an initial STNU which is not.

Case without contingent links The techniques defined can also be used in the particular case where there is no contingent link, that is where the MaSTNU is actually an MaSTN. With regards to existing work on MaSTN, one contribution is that the model introduced allows to compute robust executiong strategies which can be executed independently by the agents. We are not aware of previous works on this point for MaSTN. Also, when there is no contingent link, it can be shown that there is actually no boolean decision variable in the model and the MIP becomes a linear program solvable in polytime.

Objective functions Our method can optimize the set of local STNUs by maximizing objective function $f_{opt} = \sum_{i<j}(u_{ij} - l_{ij})$. Many other metrics can be used. For example, Fig 10 shows the local STNUs obtained when minimizing the latest execution time of the last time-point, to finish the mission as soon as possible (minimization of $f_{opt} = \max_i u_{0i}$). With this new objective function, the maximum mission completion time is reduced from 29 time units to 24 time units.
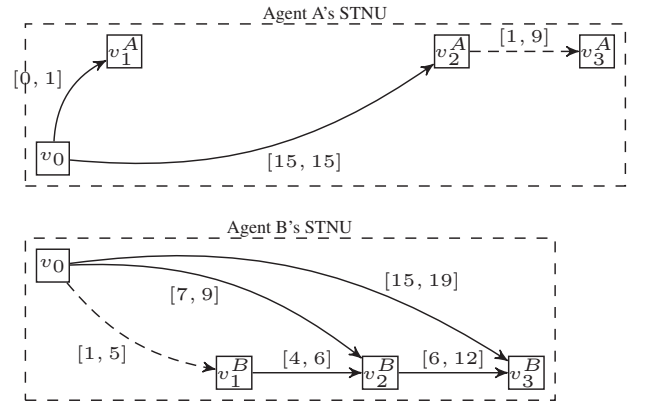

Figure 10. Mission duration optimization

Objective functions can also be used to balance the flexibility of solutions between agents, in order to avoid overly constrained agents. In this case, we define the *normalized flexibility* metrics f: $\forall a \in \mathcal{A}, f(a) = \frac{1}{|V^A|}\sum_{v_i, v_j \in V^A}(u_{ij} - l_{ij})$. The corresponding objective function to maximize is then $f_{opt} = \min_{a \in \mathcal{A}} f(a)$. This is particularly useful when the number of agents is high, since in this case the maximum global flexibility can often be reached by constraining as much as possible an unique agent. Another option can be to keep the initial objective function and constrain the problem such that each agent achieves a minimal threshold t of normalized flexibility: $\forall a \in \mathcal{A}, f(a) \ge t$.

5 Experiments

Running Times We tested our MIP approach using the CPLEX solver on 500 instances of randomly-generated MaSTNUs, ranging homogeneously from 10 to 40 nodes. All experiments were run on 3.0GHz Intel cores and 4GB memory. Finding the optimal solution to the MIP problem typically takes between 0.2 seconds for the 6-nodes example used in this paper, and 1200 seconds for a 4-agents and 40-nodes example containing 4 external contingent links and 10 external requirement links. However, in this last case a first solution, that strictly improves the solution found by removing external contingent links, is found in 80 seconds, at the expense of a drop in the flexibility of 75% compared to the optimal solution.

It must be noted that our current implementation of DC for local STNU (section2.4) is based on the $O(N^5)$-time DC-checking techniques from [9], and is the primary cause of the scalability performances. A more efficient algorithm in $O(N^4)$-time can be found in [6], however this algorithm is more complex to translate into a MIP formalism and is beyond the scope of this paper.

Impact of Observations on Temporal Flexibility For the sake of consistency with real life applications vocabulary, in the remainder of this paper we will refer to external contingent links as *observations*.

We want to measure the improvements made by our method on the "quality" of resulting execution strategies. To this end, we define the *Relative temporal flexibility* of a solution as being the ratio of the value of the objective function of solution to the value of the objective function of the corresponding STNU:

Relative Temporal Flexibility $= \frac{f_{opt(MaSTNU)}}{f_{opt(STNU)}}$.

This metric gives us a strong indicator of the "quality" of a solution compared to best and worst cases. At 100% it means the solution is as much flexible as the corresponding STNU, at 0% it means that the solution found is totally rigid with no tolerance for execution error.

We also assume that external contingent links take values in $[0, x]$, i.e observations of events by other agents are made within x unit of times. We compare x to the temporal flexibility of external requirement links in order to obtain the *relative delay of observations*:

Relative delay of observations $= \frac{|E^X|}{|C^X|} \cdot \frac{\sum_{(v_i,v_j) \in C^X} (u_{ij})}{\sum_{(v_i,v_j) \in E^X} (u_{ij}-l_{ij})}$.

This ratio measure the "quality" of observations (lower ratio means better quality), which may be translated as cost in real applications.

Fig 11 shows the impact of the number of observations (i.e number of external contingent links) and their quality (i.e their immediacy) on the relative flexibility of the solution found by our method. The parameters of the generated MaSTNUs are as follow: 4 agents of 5 nodes each, 4 external requirement links (connecting two randomly-chosen nodes from distinct agents). The number of internal requirement (resp. contingent) links for each agent were randomly drawn from 4 to 6 (resp. from 0 to 2). Additional nodes and external contingent links, representing observations, were then added depending on the experiments.

Each point on a curve represents the mean of the relative flexibility of the solution found over 10 MaSTNUs randomly-generated with the corresponding set of parameters. We display the results for two limit cases, the "Full observations" and the "No observations" cases, and an intermediary one, the "Half observations" case.

The "No observation" curve corresponds to MaSTNUs without any external contingent links: $C_X = \emptyset$. In this case, agents cannot receive any informations during execution, so the flexibility of the solution is minimal: agents must agree on a rigid schedule before
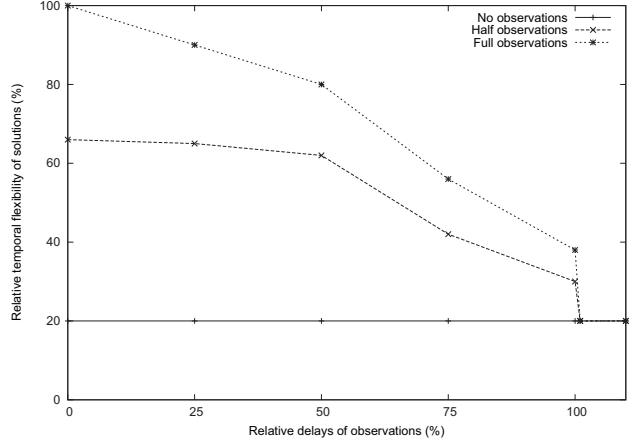

Figure 11. Influence of numbers of observations and delays

execution start. This threshold depends on the MaSTNUs considered, other MaSTNUs may have lower or higher relative flexibility than the 20% reported in our experiments in case of absence of observations.

The "Full observations" curve corresponds to MaSTNUs wherein each event is observed by each other agent: $\forall v \in V, \forall a \in A, a \neq owner(v), \exists w \in V^a$ s.t. $(v, w) \in C_X$. In the extreme case with no delays of observation, the resulting solution has the same quality than if the MaSTNU were considered as a STNU. In the opposite extreme case with high delays of observation, no useful informations can be received from the observations, and the agents must act on their own as in the no observation case.

The "Half observations" curve corresponds to MaSTNUs identical to the "Full observations" set, except half of vertices in V are not observed by any other agent. The quality of the solution actually depends on which vertices are observed: for instance observations of external vertices are more likely to be useful than observations of internal vertices.

Higher numbers of observations lead to higher flexibility of solutions, at the expense of increased computational costs and potentially increased workload during the mission if the observations were not initially scheduled.

6 Conclusion

Dynamic controllability is an important property for temporal plans with uncertainty as it improves the odds of success of a mission. In this paper we showed how to handle uncertain temporal constraints in multi-agent temporal plans thanks to Multi-agent Simple Temporal Network with Uncertainty, and how to use a MIP model to get executable plans which are dispatched between the agents. There are several future work directions for improving the management of MaSTNU, such as solving the MIP incrementally to repair infeasible MaSTNUs by adding observations one by one while optimizing computation times, or distributing the MIP solving process in order to reoptimize temporal plans during the mission, or taking into account the existence of communication rendez-vous [8].

REFERENCES

[1] James C. Boerkoel, Léon Planken, Ronald Wilcox, and Julie A. Shah, 'Distributed Algorithms for Incrementally Maintaining Multiagent Simple Temporal Networks', in *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, Rome, Italy, (2013).

[2] Guillaume Casanova, Cédric Pralet, and Charles Lesire, 'Managing dynamic multi-agent simple temporal network', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1171–1179, Istanbul, Turkey, (2015).

[3] Jing Cui, Peng Yu, Cheng Fang, Patrik Haslum, and Brian C. Williams, 'Optimising bounds in simple temporal networks with uncertainty under dynamic controllability constraints', in *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 52–60, Jerusalem, Israel, (2015).

[4] Rina Dechter, Itay Meiri, and Judea Pearl, 'Temporal Constraint Networks', *Artificial Intelligence*, **49**(1-3), 61–95, (1991).

[5] Luke Hunsberger, 'Efficient execution of dynamically controllable simple temporal networks with uncertainty', *Acta Informatica*, **52**(8), 1–59, (2015).

[6] Paul Morris, 'A structural characterization of temporal dynamic controllability', in *Principles and Practice of Constraint Programming (CP)*, pp. 375–389, Nantes, France, (2006).

[7] Paul Morris, 'Dynamic controllability and dispatchability relationships', in *Integration of AI and OR Techniques in Constraint Programming - 11th International Conference (CPAIOR)*, pp. 464–479, Cork, Ireland, (2014).

[8] Paul H. Morris and Nicola Muscettola, 'Managing temporal uncertainty through waypoint controllability', in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1253–1258, Stockholm, Sweden, (1999).

[9] Paul H. Morris, Nicola Muscettola, and Thierry Vidal, 'Dynamic Control Of Plans With Temporal Uncertainty', in *Int. Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, WA, USA, (2001).