Which Contingent Events to Observe for the Dynamic Controllability of a Plan

Arthur Bit-Monnot, Malik Ghallab, Félix Ingrand

Which Contingent Events to Observe for the Dynamic Controllability of a Plan

Arthur Bit-Monnot Malik Ghallab Félix Ingrand
LAAS-CNRS, Université de Toulouse, France
{bit-monnot, malik, felix}@laas.fr

Abstract

Planning and acting in a dynamic environment require distinguishing *controllable* and *contingent* events and checking the *dynamic controllability* of plans. Known procedures for testing the dynamic controllability assume that all contingent events are observable.

Often this assumption does not hold. We consider here the general case of networks with *invisible* as well as *observable* contingent points. We propose a first procedure for testing their dynamic controllability. Further, we define an algorithm for choosing among the observable contingent points which to observe with additional sensing actions, such as to make a plan dynamically controllable. We show how these procedures can be incrementally integrated into a constraint-based temporal planner.

1 Introduction

The issue of *controllable* vs *contingent* events in temporal reasoning problems has been recognized for some time. It triggered a long line of research that matured recently into scalable consistency and controllability checking algorithms.

A controllable timepoint is a free variable that can be set by the agent to meet its constraints. A contingent timepoint refers to an event not under the agent control; it is a random variable whose value is *observed* when that event occurs. Constraints on contingent points express knowledge about the possible range of these random variables. A Simple Temporal Network with Uncertainty (STNU) extends the usual STN model to handle both types of points. An STNU is *Dynamically Controllable* (DC, or controllable for short) if there is an execution strategy that allows to set values to future controllable points, given the observation of past contingent points.

Results and algorithms developed for STNUs rely on the assumption that all contingent points are *observable*. In many cases this assumption is questionable. Contingent points result from the proper dynamics of the world. The agent can make enough prediction about this dynamics to plan its activity. However, it may or may not be able to observe precisely the occurrence of the predicted contingent points. It may have to deal with events whose moments of occurrence are invisible. For instance, synchronizing one's activity with a distant partner may involve timepoints that are invisible (Figure 1). Further, there may be contingent points that remain observable, but that are hidden unless specific actions are taken. This is essential for planning in a dynamic domain. Given the planned activity with respect to a predicted contingent evolution of the environment, the agent has to decide what it needs to observe to make its plan controllable and what observation actions are consistent with its activity.

The focus of this paper is not on planning *per se*, but on the controllability of Partially Observable STNUs (POSTNUs). We propose the following contributions:

- We partition contingent points into *invisible* and *observable* ones. We define a procedure called PODC that maps a POSTNU into an equivalent STNU, and demonstrate that when the latter is DC then the POSTNU is

Acknowledgement: This work was supported in part by the EU MUMMER project funded by the H2020 program under grant agreement No 688147 and the EDSYS Doctoral School of the University of Toulouse.

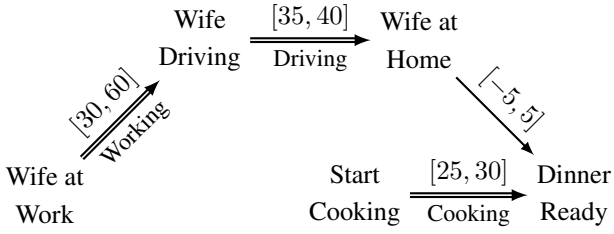

Figure 1: This simple example illustrates *dynamic controllability*: the STNU is controllable only if the event *Wife Driving* is observed. This observation may require sensing actions.

also DC with respect to its observable points. To our knowledge, PODC is the first DC-checking algorithm for POSTNUs.

- We further partition the *observable* timepoints into *visible* and *hidden* ones, the latter require observation actions to become visible to the agent (Figure 4). We define an algorithm called NEEDEDOBS for choosing among the hidden points what to observe to make the POSTNU controllable. NEEDEDOBS relies on PODC and on an edge labeling mechanism whose main property is proved. We report on the performance of PODC and NEEDEDOBS.
- We show how the above two procedures can be used *incrementally* in temporal planning and illustrate their integration into a constraint-based temporal planner.

These three issues are covered respectively in Sections 3, 4 and 5, preceded by a discussion of the state of the art.

2 Related Work

Barták et al. (2014) survey the state of the art in temporal networks. STNUs were introduced by Vidal and Ghallab (1996). Strong, weak and dynamic controllability cases were analyzed by Vidal and Fargier (1999). Algorithms for strong and weak controllability were proposed by Cimatti et al. (2012a,b). State space planning with strong controllability is studied by Cimatti et al. (2015). Several extensions of the STNU model have been proposed, e.g., with conditions and disjunctions Cimatti et al. (2014); Hunsberger et al. (2015).

A polynomial algorithm for the dynamic controllability of STNUs was proposed by Morris et al. (2001), improved by Morris and Muscettola (2005), Morris (2014) and Hunsberger (2015). The latter two algorithms runs in $O(n^3)$. An implementation of Morris (2014) supports our results.

Incremental algorithms for checking the dynamic controllability of STNUs have been introduced by Stedl and Williams (2005) and Shah et al. (2007). They have been improved by Nilsson et al. (2013, 2014); their latest version runs in $O(n^3)$.

Dynamic execution strategies for triggering online controllable points in a DC STNU are studied by Tsamardinos et al. (1998) and Morris (2014).

Partially Observable STNUs (POSTNU) were introduced by Moffitt (2007) together with an incomplete filtering algorithm. If that filter returns an affirmative output then its input POSTNU is *not* DC, otherwise there is no guarantee that the network is dynamically controllable. To our knowledge, no other work covers Partially Observable STNUs.

3 Checking the controllability of a POSTNU

3.1 Background

An STNU extends an STN with controllable and contingent nodes. It has two types of links:

- *requirement links*, denoted $A \xrightarrow{[l,u]} B$, specify that point B can be chosen anywhere in the interval $[A + l, A + u]$.
- *contingent links*, $A \xRightarrow{[l',u']} C$, say that *contingent* node C will occur at a random point in $[A + l', A + u']$.

An STNU is *Controllable* (DC for short)[1] if there is an execution strategy that chooses future controllable points given the observation of past contingent ones, while meeting all the constraints. A necessary but not sufficient condition for controllability requires the network to be consistent and the minimal consistent network should not reduce any contingent link.

[1] Unless stated otherwise, *controllable* refers to dynamic controllability.

Conditions	Added constraint
$A \xleftarrow{B:x} C \xleftarrow{y} D$	$A \xleftarrow{B:(x+y)} D$
$A \xleftarrow{x} C \xleftarrow{c:y} D,\ x < 0$	$A \xleftarrow{x+y} D$
$A \xleftarrow{B:x} C \xleftarrow{c:y} D,\ x < 0, B \neq C$	$A \xleftarrow{B:(x+y)} D$
$A \xleftarrow{x} C \xleftarrow{y} D$	$A \xleftarrow{x+y} D$
$B \xleftarrow{b:x} A \xleftarrow{B:z} C,\ z \geq -x$	$A \xleftarrow{z} C$

Table 1: Constraint propagation rules in an STNU

A *labeled distance graph* representation is convenient for checking the controllability of STNUs. We use it in the rest of this paper. It relies on transforming the STNU network into a labeled multigraph obtained as follows. Each requirement link $A \xrightarrow{[x,y]} B$ is replaced by two edges $A \xrightarrow{y} B$ and $A \xleftarrow{-x} B$. A contingent link $A \xRightarrow{[x,y]} B$ is replaced by four edges: the same two edges as for a controllable link, and two labelled edges $A \xrightarrow{b:x} B$ and $A \xleftarrow{B:-y} B$, called *lower-case* and *upper-case* edges, referring respectively to the lower and upper bound values of the contingent link.

A labeled distance graph can be used to compute distances between nodes, as in a distance graph of an ordinary STN. The latter is consistent if and only if its distance graph does not contain a negative cycle. Specific propagation rules (Table 1) have been devised for labelled distance edges to provide a similar property: an STNU is controllable iff it does not have a so-called *semi-reducible negative cycle* obtained with these constraint propagation rules Morris (2014). Procedure DC-CHECK takes as input an STNU; it propagates these constraints with a Dijkstra-like algorithm; it returns *true* when no such a negative cycle is found, which entails a DC STNU.

3.2 Transforming a POSTNU into an STNU

A POSTNU is an STNU where contingent events are partitioned into *observable* and *invisible* events.

Definition 1. A POSTNU is a tuple $\Omega = (X_C, X_I, X_O, C)$ where:

- X_C is a set of controllable timepoints;
- X_I is a set of *invisible* contingent events;
- X_O is a set of *observable* contingent events;
- C is a set of requirement and contingent links.

A POSTNU is controllable if there exists an execution strategy that chooses future controllable points given the observation of past *observable* points. If there is no invisible event, then a POSTNU is simply an STNU. If there is no observable event, then a POSTNU is DC iff the corresponding STNU is *strongly* controllable, i.e., there are values of X_C that meet all the constraints regardless of the values of X_I.

Conditions	Added constraint
$A \xleftarrow{B:x} B \xleftarrow{y} C$	$A \xleftarrow{x+y} C$
$A \xleftarrow{B:x} B \xleftarrow{C:y} C$	$A \xleftarrow{C:x+y} C$
$A \xleftarrow{x} B \xleftarrow{b:y} C$	$A \xleftarrow{x+y} C$
$A \xleftarrow{a:x} B \xleftarrow{b:y} C$	$A \xleftarrow{a:x+y} C$

Table 2: Added constraints due to an invisible point B.

In order to check the controllability of POSTNU we map it into an STNU. The mapping removes invisible nodes and adds constraints on controllable and observable points, as specified in Table 2. The controllability of the resulting STNU is then checked.

This is performed by procedure PODC (Figure 2). For every invisible node x_I, it propagates the constraints in Table 2 (PROPAGATEINVISIBLE) and it removes all edges to and from x_I (REMOVEEDGES). At that point (X_C, X_O, C') is an STNU whose controllability is tested with DC-CHECK.

```
PODC(X_C, X_I, X_O, C)
  C' ← C
  for each x_I ∈ X_I do
      C' ← PROPAGATEINVISIBLE(C', x_I)
      C' ← REMOVEEDGES(C', x_I)
  return DC-CHECK(X_C, X_O, C')
```

Figure 2: Maps a POSTNU into a fully observable STNU whose controllability is tested with DC-CHECK.

Proposition 1. *If* PODC(Ω) *returns true, then the POSTNU* $\Omega = (X_C, X_O, X_I, C)$ *is dynamically controllable.*

Proof. Consider an invisible timepoint B with an incoming contingent link $A \xRightarrow{[l,u]} B$ (i.e., $A \xrightarrow{b:l} B$ and $A \xleftarrow{B:-u} B$ in the labeled distance graph). B cannot be the target of more than one contingent link, otherwise the network is not DC. There are three possible constraints that may involve B:

- B is the target of a requirement edge $B \xleftarrow{-y} C$, which says *"C must occur at least y time units after B"*. The first rule in Table 2 adds the edge $C \xleftarrow{-y-u} A$. This edge dominates the original one: a minimum delay of at least y will be maintained between B and C regardless of the outcome of the contingent link $A \Rightarrow B$.
- B is the source of a requirement edge $B \xrightarrow{x} C$, which says *"C should occur at most x time units after B"*. The third rule in Table 2 introduces an edge $A \xrightarrow{x+l} C$. The original edge is dominated by this edge: a maximum delay of x is enforced between B and C regardless of the outcome of the contingent link $A \Rightarrow B$.
- B is the source of a contingent link $B \xRightarrow{[x,y]} C$, giving two edges $B \xrightarrow{c:x} C$ and $B \xleftarrow{C:-y} C$. The second and fourth rules in Table 2 add the edges $A \xrightarrow{c:x+l} C$ and $A \xleftarrow{C:-y-l} C$. These merge the two contingent links $A \Rightarrow B \Rightarrow C$ into one, ignoring B as an intermediary point. They dominate the original ones as they are less informative.

Consequently, any edge involving B in the original POSTNU is dominated by an edge introduced by the rules in Table 2. All those edges to and from B can thus be removed from the STNU without relaxing the problem. □

The converse of Proposition 1 does not hold in general, since the transformation of PODC is conservative on some POSTNUs. An example of such network is given in Figure 3. This network has a *chained contingency*: there is a contingent timepoint B that is at the center of a contingent chain $A \Rightarrow B \Rightarrow C$ and is involved in an additional contingent or requirement link with another timepoint D.

The converse of Proposition 1 holds for a POSTNU without chained contingencies. Let us focus on this class of networks; we discuss later why it is relevant in planning.

Proposition 2. *A POSTNU Ω with no chained contingencies is controllable if and only if* PODC(Ω) *holds.*

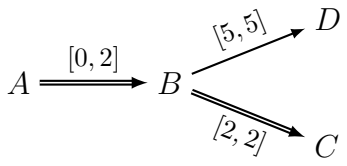

Figure 3: A POSTNU where PODC is too conservative. The network is controllable even if B is invisible: just schedule D 3 units after C. Here, B is *indirectly observable* through C.

Proof (sketch). It is sufficient to show that, for given contingent point B, either *(i)* no indirect information can be gathered on B after its occurrence; or *(ii)* such information is useless since B is not involved in any requirement link. □

The complexity of PODC is in $O(n^3)$ for a network of n nodes: the *for* iteration is in $O(|X_I| \times |C|)$ dominated by the DC-CHECK step of cubic complexity.

4 Finding what to observe in a POSTNU

We partitioned contingent events into invisible and observable ones. However, an observable event may or may not be visible to an agent depending on, e.g., its location, the setting of its sensors or its concurrent activity. The agent may need to perform specific actions to perceive observable events.

Given a plan π, we further partition the set of *observable* contingent events X_O into two subsets (Figure 4):

- $X_V(\pi)$: events whose occurrence will always be *visible* to the agent, e.g., when the phone in my pocket rings;
- $X_H(\pi)$: events that are normally *hidden*, they become visible if specific actions are added to π, e.g., to observe that the water boils I need to be close and pay attention to it.

Contingent events in a plan π are: $X_I \cup X_V(\pi) \cup X_H(\pi)$. Invisible events are intrinsic to a domain model, which has no means to observe them. However, the partition of observable events depends on the specification of a planning problem as well as on the particular plan for

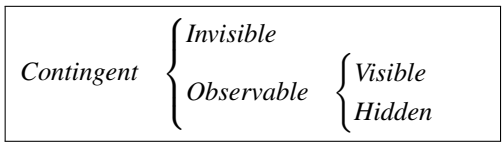

Figure 4: Classes of contingent points.

that problem: an event initially in X_H migrates to X_V iff the actions needed to observe it are added to π. To make a plan controllable, an agent may need to perform additional sensing actions.

4.1 Needed Observations

Let $\Omega_\pi = (X_C, X_I, X_V(\pi), X_H(\pi), C)$ be a POSTNU without chained contingencies corresponding to a plan π. $X_V(\pi)$ are the events initially specified as visible and those hidden that become visible because of the activity already planned in π. Ω_π can be in one of the following three cases:

(1) PODC$(X_C, X_I, X_V(\pi) \cup X_H(\pi), C)$ is false: π cannot be made controllable even if all $X_H(\pi)$ is observed.

(2) PODC$(X_C, X_I \cup X_H(\pi), X_V(\pi), C)$ is true: π is controllable without any additional observation.

(3) Otherwise additional observation actions of events in $X_H(\pi)$ may make π controllable.

The third case requires identifying in X_H events whose observation makes the augmented plan controllable. This is performed with procedure NEEDEDOBS (Figure 5), which searches a space of subsets of X_H. A path in the search tree corresponds to a set of events to move from the initial X_H to X_V. A search state is expanded by nondeterministically choosing an event x to observe in a set of candidates. This is a backtrack point, but the order in which the candidates are examined is irrelevant. The search fails when no candidates can be found in X_H. It succeeds in finding a set of observations, *minimal* in the set inclusion sense, that make the network DC.

OBSCANDIDATES is a key function in NEEDEDOBS: it selects a subset of X_H whose observation might render the POSTNU controllable. A naive version would simply try all possibly hidden events. Let us discuss how to find a focused set of candidates.

NEEDEDOBS(X_C, X_I, X_V, X_H, C)
  if PODC$(X_C, X_I \cup X_H, X_V, C)$ then
    return $\varnothing$
  $\Sigma \leftarrow$ OBSCANDIDATES(X_H)
  if $\Sigma = \varnothing$ then return *failure*
  $x \leftarrow$ NONDETERMINISTICALLYCHOOSE(Σ)
  $\sigma \leftarrow$ NEEDEDOBS$(X_C, X_I, X_V \cup \{x\}, X_H \setminus x)$
  if $\sigma =$ *failure* then return *failure*
  else return $\{x\} \cup \sigma$

Figure 5: Finding a set of observations to make a Partially Observable STNU Dynamically Controllable

4.2 Finding relevant candidates for observation

The key insight to focus the search is to analyze what makes a given POSTNU not DC. We first remark that the transformations in Table 2 simply make sure that the network stays consistent regardless of whether an invisible node occurs at its earliest or at its latest. More specifically, the first and second rules propagate the upper-case edges, while the last two rules propagate the lower-case edges of the invisible node.

We say that an edge e enforces the upper bound of a invisible point B if it was either: *(i)* introduced by the first or second rule of Table 2 with B as an invisible point; or *(ii)* derived through constraint propagation from an edge enforcing the upper bound of B. Similarly for the lower bounds with respect to the last two rules. We say that an invisible point is enforced by a sequence of edges if at least one edge of the sequence enforces that point.

We make this information explicit in the network by labeling all edges with the set of points they enforce. For instance, if an edge is labeled with the set $\{\overline{A}, \overline{B}, \underline{B}\}$, it means it enforces the upper bounds of A and B (noted $\overline{A}$ and $\overline{B}$), and the lower bound of B (noted $\underline{B}$).

Procedure PODC is extended to initialize these labels. Table 3 shows the extended transformations with the additional labels (in red). For an invisible event B, the added constraint inherits the label of the two constraints it orig-

inates from (the sets U and V) and has an additional projection $\underline{B}$ (resp. $\overline{B}$) if it enforces the lower (resp. upper) bound of B.

Conditions	Added constraint
$A \xleftarrow{B:x,\ U} B \xleftarrow{y,\ V} C$	$A \xleftarrow{x+y,\ U\cup V\cup\{\overline{B}\}} C$
$A \xleftarrow{B:x,\ U} B \xleftarrow{C:y,\ V} C$	$A \xleftarrow{C:x+y,\ U\cup V\cup\{\overline{B}\}} C$
$A \xleftarrow{x,\ U} B \xleftarrow{b:y,\ V} C$	$A \xleftarrow{x+y,\ U\cup V\cup\{\underline{B}\}} C$
$A \xleftarrow{a:x,\ U} B \xleftarrow{b:y,\ V} C$	$A \xleftarrow{c:x+y,\ U\cup V\cup\{\underline{B}\}} C$

Table 3: Added constraints due to an invisible point B, with the labels propagated to track the points enforced by edges.

We also extend the classical STNU reduction rules of Table 1 as follow: if a reduction is triggered that produces an edge e_3 from two edges e_1 and e_2, then e_3 is annotated with the labels of both e_1 and e_2. Let φ be an STNU issued from the transformation by Table 3 of a POSTNU. If φ is not DC then it necessarily has a "culprit" sequence of edges (a semi-reducible negative cycle). This sequence, denoted φ_{Culprit}, enforces invisible and hidden nodes in X_I and X_H; in the latter case it can give us observation candidates.

Proposition 3. *Let $\Omega = (X_C, X_I, X_V, X_H, C)$ be a POSTNU without chained contingencies such that $\text{PODC}(X_C, X_I \cup X_H, X_V, C)$ is false, and φ be the corresponding STNU. If there is a sequence $\varphi_{Culprit}$ that enforces both the upper and lower bounds of events in X_H, then the observation of at least one such event is needed to make Ω controllable. Otherwise Ω cannot be made controllable.*

Proof. Making Ω controllable requires a change in φ in order to remove φ_{Culprit}. The only allowed changes result from observing events in X_H. Observing an event A means that edges labeled with $\underline{A}$ or $\overline{A}$ (and only those) won't be introduced in φ. Since the only edges that can be removed are the ones labeled with a node in X_H, then if there is none (φ_{Culprit} enforces only nodes in X_I) φ remains non DC regardless of additional observations. Otherwise the observation of at least one event in X_H enforced by the sequence φ_{Culprit} is needed.

Let φ' be a partial projection of φ defined as follows: for every contingent link $A \xRightarrow{[l,u]} B$ where B has its lower (resp. upper) and only its lower (resp. upper) bound enforced by φ_{Culprit}, we replace in φ' this link by a new link $A \xRightarrow{[l,l]} B$ (resp. $A \xRightarrow{[u,u]} B$). φ' is essentially a version of φ where the duration of all contingent links with only one bound enforced by φ_{Culprit} is known in advance.

The main feature of φ' is that it contains all edges of φ_{Culprit}, making φ' not DC. In φ', complete knowledge of events with just one enforced bound is not sufficient to remove the inconsistent sequence. Consequently, removing the inconsistency requires at least one event with its two bounds participating in φ_{Culprit} to be observed. □

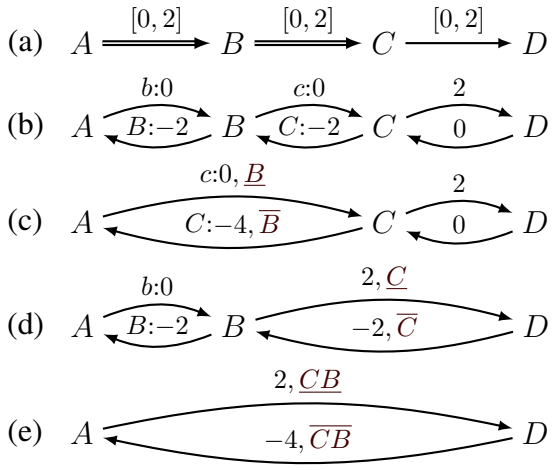

Figure 6: (a) A POSTNU with two contingent points B and C; its labelled distance graph (where requirement edges are omitted for clarity) in different observability cases: (b) both B and C are observable, (c) B is invisible, (d) C is invisible, (e) both B or C are invisible.

The OBSCANDIDATES function returns $\varnothing$ when there is a sequence φ_{Culprit} (obtained when PODC is false) which does not enforce the upper and lower bounds of a node in X_H. Otherwise it returns all such nodes.

Figure 6(e) gives an example of such an inconsistent network. It was built by applying PODC to the network of Figure 6(a) with B and C marked invisible. The edge $A \xrightarrow{2,\ \underline{CB}} D$ states that the lower bounds of both B and C require a delay from A to D of at most 2. The edge $A \xleftarrow{-4,\ \overline{CB}} D$ states that the upper bounds of both B and C require a delay from A to D of at least 4. These

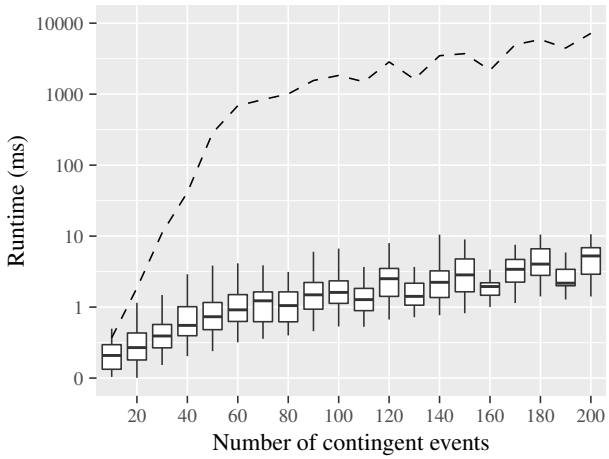

Figure 7: NEEDEDOBS runtime distribution on 2264 random networks of 32 to 311 events in total, requiring 1 to 4 observations to be DC (median, first and third quartiles, max and min); dashed line: gives the unfocused search median runtime.

two edges will be detected as a culprit cycle that enforces both the lower and upper bounds of B and C. To make the network controllable, at least one of B or C must be observed. This would result in the STNUs of either Figures 6(c) and 6(d) which are both controllable.[2]

Figure 7 shows an empirical evaluation of the performance of NEEDEDOBS on randomly generated POSTNUs corresponding to typical planning domains with a few hundred nodes and up to 200 contingent events.[3] The labeling technique speeds up the search by several orders of magnitude over the naïve approach. Indeed, NEEDEDOBS required on average 5 iterations, and as many calls to DC-CHECK, to find a minimal set of observations. Even on large networks, this number never exceeded 13.

5 Planning with a POSTNU

Our proposed approach and procedures PODC and NEEDEDOBS can be integrated incrementally into a temporal planner. Let us describe how this was done with FAPE Dvořák et al. (2014), a constraint-based hierarchical temporal planner for the ANML language of Smith et al. (2008).

FAPE searches a space of partial plans by fixing flaws in a current partial plan until no flaw remains. Flaw are for example unachieved goals, unrefined tasks or threads to the consistency or causal relations of the plan.

To handle POSTNUs we need a way to refer to contingent events and their sensing actions. We associate to every observable contingent event x a boolean state variable sv_x that is true when x can be observed. A sensing action for x is any action that makes sv_x true, before and during the entire interval of the expected occurrence time of x. An observable event x is initially in X_V if sv_x is unconditionally always true.

Let $\Omega_\pi = (X_C, X_I, X_V(\pi), X_H(\pi), C)$ be the POSTNU corresponding to a plan π as defined in subsection 4.1. A contingent event x is: *(i)* in X_I if sv_x is always false and there is no sensing action for x; *(ii)* in $X_V(\pi)$ if sv_x is true over the interval of occurrence of x; *(iii)* in $X_H(\pi)$ otherwise.

Controllability checks are incrementally integrated in the planning procedure. When a partial plan π is selected for expansion, NEEDEDOBS is run to determine whether π requires additional observation to be made controllable. If it does, a new flaw ϕ is introduced for π. ϕ has one resolver for every minimal set of observations given by NEEDEDOBS. This resolver contains, for every event x in the observation set, an assertion requiring sv_x to be true over the timespan of x.

When an assertion is added to a partial plan, an *open goal* flaw is created: either *(i)* the assertion already holds or can be made to hold with temporal or binding constraints, or *(ii)* the assertion does not hold. The latter is handled by introducing a new enabling action, in our case a sensing action, if one exists. Hence, the resolver ensures that all necessary sensing actions end up in the plan. If multiple sensing actions are available for an event, the planner's search will branch on this choice, as when solving an open goal.

This procedure adds incrementally the required sensing actions while building the plan; it permits early backtracks when dead-ends are encountered. For the sake of completeness, NEEDEDOBS is extended to give another set of observations if the previous ones cannot be added to the plan.

[2] Note that the network in Figure 6 has no chained contingencies.

[3] All tests were run on an Intel i7 processor with 4 GB RAM.

Let us go back to the issue of chained contingencies. Contingent points in a plan are usually the end points or the intermediate points of the agent own actions as well as the expected events to be triggered by the environment independently of its actions. The expectations regarding the latter are often known with respect to absolute time (e.g., periodic events), which rule out chained contingencies. Expected events, supposed to occur regardless of the agent activity, can seldom be defined with respect to the end points of its actions. Further, two consecutive actions cannot make a chained contingency. Remains as a case of such contingencies intermediate and final points of actions. This case can be avoided by decomposing the action or expressing the contingent constraints with respect to the controllable point. If this is not convenient, then NEEDEDOBS loses completeness but remains sound and useful in practice. Finally let us note that for a network with chained contingencies, if PODC is false even when all observables are assumed visible (case 1 in subsection 4.1), then one can apply the filtering procedure of Moffitt (2007) to possibly confirm that this network is not controllable; but a negative output leaves the issue open.

A full presentation of FAPE and its performance is not within the scope of this paper. However, to test our approach beyond the results on random POSTNUs, we run it on FAPE on a simple logistics domain that has n areas. In each area several contingent events of different types occur. An event e produces an effect that holds for a given amount of time. An agent can: *(i)* move between the areas, *(ii)* observe the occurrence of events in its area, and *(iii)* handle an event by acting during the period its effect holds. Each problem instance has two agents and few tasks corresponding to events that need be handled.

The planner is run on 192 randomly generated instances of this domain. The typical solution involves a few move actions of the agents and one handling action per task. A handled event leads either to: *(i)* add a sensing action to observe this event; *(ii)* add one or several sensing actions to observe earlier events to provide enough information; or *(iii)* no sensing action is needed. The planner is able to identify which of these options is possible and choose a desirable one appropriately.

Figure 8 shows the runtime distribution to plan problems having different number of tasks. More tasks result in more actions in the final plan, which naturally increases planning time. The planner spent on average 47.31%, and

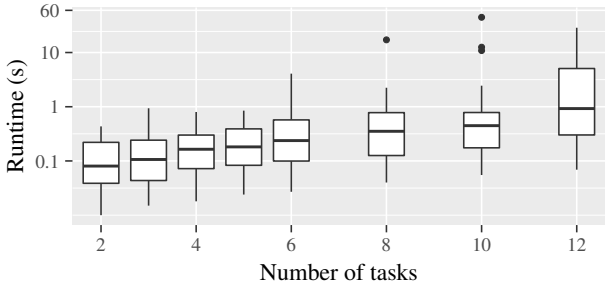

Figure 8: FAPE runtime distribution on 192 random instances of a logistics domain with 8 to 200 contingent events (isolated points are beyond the maximum plotted by R).

at most 68%, of its processing time for identifying necessary observations. Three problems were not solved for a timeout of 60 seconds. On those problems, the limitation was the heuristic of the planner that failed to provide sufficient guidance, since our usual heuristics are not designed to assess the specifics of partial observability. Beyond the quantitative measures, it is important to note that, with a reasonable overhead, the planner was able to incrementally identify the necessary observations to keep its plan dynamically controllable. More importantly, this was done while maintaining and extending the causal structure of the plan to support the required sensing actions.

6 Conclusion

This paper presents a fairly comprehensive study of temporal networks with partially observable contingent events. The dynamic controllability of STNUs assumes all contingent events to be observable. In many domains they are not. However, the agent can observe them if it takes adequate sensing actions, hence if it plans for those actions.

We proposed a transformation of POSTNUs into STNUs to determine their controllability using known DC-CHECK algorithms. We formally proved PODC to be sound; it is complete for an interesting subclass of networks. This advances the state of the art with respect to the other known algorithm Moffitt (2007) which cannot guarantee that a network is controllable.

Building on PODC, we proposed NEEDEDOBS for finding a minimal set of observations that makes a POSTNU controllable. We developed an edge labelling mechanism that focuses NEEDEDOBS and demonstrated formally its main property. Empirical tests show that NEEDEDOBS remains efficient even on networks corresponding to large plans.

We showed how the proposed approach can be integrated incrementally to a temporal planner. The required extensions to a constraint-based hierarchical planner were briefly discussed, together with empirical results of our implementation in a domain designed for testing our POSTNU approach.

Finally, we believe that this paper opens an interesting research perspective regarding partial observability, beyond our present focus on its use in temporal plans. Indeed, classical models of partially observable dynamic systems consider only two categories of variables: invisible and observable. Our partition in three categories is essential in most domains, since visibility is also a dynamic property.

References

Barták, R., Morris, R., and Venable, B. (2014). *An Introduction to Constraint-Based Temporal Reasoning*. Morgan & Claypool.

Cimatti, A., Hunsberger, L., Micheli, A., Posenato, R., and Roveri, M. (2014). Sound and Complete Algorithms for Checking the Dynamic Controllability of Temporal Networks with Uncertainty, Disjunction and Observation. In *International Symposium on Temporal Representation and Reasoning (TIME)*, pages 27–36.

Cimatti, A., Micheli, A., and Roveri, M. (2012a). Solving temporal problems using SMT: Strong controllability. In *Proc. Int. Conf. Principles and Practice of Constraint Programming (CP)*.

Cimatti, A., Micheli, A., and Roveri, M. (2012b). Solving temporal problems using SMT: Weak controllability. In *Proc. AAAI*.

Cimatti, A., Micheli, A., and Roveri, M. (2015). Strong temporal planning with uncontrollable durations: a state-space approach. In *Proc. AAAI*, pages 1–7.

Dvořák, F., Barták, R., Bit-Monnot, A., Ingrand, F., and Ghallab, M. (2014). Planning and Acting with Temporal and Hierarchical Decomposition Models. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 115–121.

Hunsberger, L. (2015). New techniques for checking dynamic controllability of simple temporal networks with uncertainty. In *Agents and Artificial Intelligence*, pages 170–193.

Hunsberger, L., Posenato, R., and Combi, C. (2015). A Sound-and-Complete Propagation-Based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks. In *International Symposium on Temporal Representation and Reasoning (TIME)*, pages 4–18.

Moffitt, M. D. (2007). On the Partial Observability of Temporal Uncertainty. In *Proc. AAAI*, pages 1031–1037.

Morris, P. (2014). Dynamic Controllability and Dispatchability Relationships. In *Integration of AI and OR Techniques in Constraint Programming*, volume 8451, pages 464–479.

Morris, P. and Muscettola, N. (2005). Temporal Dynamic Controllability Revisited. In *Proc. AAAI*, pages 1193–1198.

Morris, P., Muscettola, N., and Vidal, T. (2001). Dynamic control of plans with temporal uncertainty. In *Proc. IJCAI*, pages 494–502.

Nilsson, M., Kvarnström, J., and Doherty, P. (2013). Incremental Dynamic Controllability Revisited. In *Proc. ICAPS*, pages 337–341.

Nilsson, M., Kvarnström, J., and Doherty, P. (2014). Incremental dynamic controllability in cubic worst-case time. In *Int. Symp. on Temporal Representation and Reasoning (TIME)*.

Shah, J. A., Stedl, J., Williams, B. C., and Robertson, P. (2007). A Fast Incremental Algorithm for Maintaining Dispatchability of Partially Controllable Plans. In *Proc. ICAPS*, pages 296–303.

Smith, D. E., Frank, J., and Cushing, W. (2008). The ANML language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Stedl, J. and Williams, B. (2005). A fast incremental dynamic controllability algorithm. In *Proc. ICAPS Wksp. on Plan Execution*.

Tsamardinos, I., Muscettola, N., and Morris, P. (1998). Fast transformation of temporal plans for efficient execution. In *Proc. AAAI*.

Vidal, T. and Fargier, H. (1999). Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Experimental & Theoretical Artificial Intelligence*.

Vidal, T. and Ghallab, M. (1996). Dealing with uncertain durations in temporal constraints networks dedicated to planning. In *Proc. ECAI*, pages 48–52.