Dynamic Sequencing of Tasks in Simple Temporal Networks with Uncertainty

Thierry Vidal Julien Bidot

LGP / ENIT
47, av d'Azereix - BP 1629
F-65016 Tarbes cedex, France
thierry@enit.fr

Abstract

Planning or scheduling systems that handle tasks with uncertain durations might use an extension of the Simple Temporal Network with a distinction between controllable and contingent variables and constraints. Temporal consistency is then redefined in terms of Dynamic Controllability, which means the ability to decide the precise timing of tasks only at execution time, depending on observations made, and still satisfying all no constraints. This property has been recently proven to be checkable in polynomial time through a simple path consistency -like algorithm. In this paper, we are interested in using such a model in scheduling applications, in which tasks may compete for the same resource, and should thus be sequenced. Such constraints make the problem NP-hard, and cannot be directly expressed in a STN. In the presence of uncertainty, one might also wish to postpone task sequencing until execution time. This paper provides the characterization of such a Dynamic Sequencing ability. Then we propose an incomplete checking method still relying on the STNU for the sake of temporal reasoning efficiency, adding further filtering techniques to account for sequencing constraints.

1 Introduction

Simple Temporal Networks [9] have proved useful in planning and scheduling applications that involve quantitative time constraints (e.g. [8, 6]) because they allow fast checking of temporal consistency. However this formalism does not adequately address an important aspect of real execution domains: the time of occurrence of some events may not be under the complete control of the execution agent. For example, on a building site, a task might wait for a supply truck, which arrival time is dependent on the traffic, while the task duration itself might depend on the wheather conditions. In cases like this, the execution agent does not have freedom to select the precise time delay between events. Instead, the value is selected by Nature independently of the agent's choices. This can lead to constraint violations during execution even if the

Simple Temporal Network appeared consistent at plan generation time.

The problem of constraint satisfaction for STN with Uncertainty was addressed formally in [13]. In this setting, the question of temporal feasibility goes beyond mere consistency to encompass issues of "controllability." Essentially, a network is controllable if there is a strategy for executing the timepoints under the agent's control that satisfies all requirements, including those involving the uncontrolled timepoints. Three primary levels of controllabilityhave been identified. In *Strong Controllability*, there is a static control strategy that is guaranteed to work in all situations. In *Weak Controllability*, for all situations there is a "clairvoyant" strategy that works if all uncertain durations are known when the network is executed. The most interesting controllability property from a practical point of view is *Dynamic Controllability*, where it is assumed that each uncertain duration becomes known (is observed) just after it has finished, and it requires that an execution strategy depend only on the past outcomes.

Algorithms have been presented for checking these properties : Strong Controllability has been shown to be tractable, while Weak Controllability is co-NP-complete [13, 5]. Recently [7] Dynamic controllability was proven to be also tractable, through the application of a mere local consistency filtering algorithm. These results will be reviewed in section 2.

Actually the Dynamic Controllability (checked off line) means the ability to postpone effective timing of tasks until actually executing them on line, but resting assured that no constraint will ever be violated, whatever the observations are. This can be viewed as a least-committment approach adding flexibility to the planning and execution loop, still ensuring the plan safe execution.

In scheduling, two tasks may compete for the same resource. Then one must come before the other, which is called a sequencing decision. For example, the same crane might be needed for two unloading tasks that should be both executed within a given time window. Usually, when executing a STN, such decisions have already been made (and the added precedence link proven consistent), since a constraint such as $task_i$ *before or after* $task_j$ cannot be expressed through simple binary constraints between time-points, as it will be developped in section 3. Nevertheless, in domains with temporal uncertainties, it might be the case that one sequencing choice is compulsory to make the network Dynamically Controllable in some situations, while the reverse choice is needed in other situations, none being valid in all situations. This calls for Dynamic Sequencing strategies, which means postponing such decisions till execution time. We would then rather be able to check off line that this might be done on line without constraint violations. Such a new property will be called DS-ability (for Dynamic Sequencing ability) and will be characterized in section 4 together with Strong and Weak counterparts. Then in section 5 we will propose a filtering algorithm which is incomplete but might help proving the DS-ability of STNU. We will conclude with a few words on future extensions of this work.

2 Background on STN, STNU and Controllability

A Simple Temporal Network (STN) is a graph in which the edges are labelled with upper and lower numerical bounds. The nodes in the graph represent *timepoints*, while

the edges correspond to constraints on the durations between the events. Formally, an STN may be described as a 4-tuple $< N, E, l, u >$ where N is a set of nodes, E is a set of edges, and $l : E \rightarrow \mathbb{R} \cup \{-\infty\}$ and $u : E \rightarrow \mathbb{R} \cup \{+\infty\}$ are functions mapping the edges into extended Real Numbers, that are the lower and upper bounds of the interval of possible durations. Each STN is associated with a *distance graph* [9] derived from the upper and lower bound constraints. An STN is consistent if and only if the distance graph does not contain a negative cycle, which can be determined by a local shortest path propagation algorithm. To avoid confusion with edges in the distance graph, we will refer to edges in the STN as *links.*

A Simple Temporal Network With Uncertainty (STNU) is similar to a STN except the links are divided into two classes, *contingent links* and *requirement links*. Contingent links may be thought of as representing causal processes of uncertain duration; their finish timepoints, called *contingent* timepoints, are controlled by Nature, subject to the limits imposed by the bounds on the contingent links. All other timepoints, called *executable* timepoints, are controlled by the agent, who has to satisfy the bounds on the requirement links. We assume the durations of contingent links vary independently.

Thus, an STNU is a 5-tuple $\Gamma = < N, E, l, u, C >$, where N, E, l, u are as in an STN, and C is the subset of the contingent links. We require $0 < l(e) < u(e) < \infty$ for each contingent link e.

An STNU may be regarded as a family of STNs : a *projection* [13] of Γ is an STN derived from Γ where each requirement link is replaced by an identical STN link, and each contingent link e is replaced by an STN link with equal upper and lower bounds $[b, b]$ for some b such that $l(e) \leq b \leq u(e)$.

Given a fixed STNU $< N, E, l, u, C >$, a *schedule* T is a mapping $T : N \rightarrow \mathbb{R}$ where $T(x)$ (sometimes written T_x) is called the *time* of time-point x. A schedule is *consistent* if it satisfies all the link constraints. From a schedule, we can determine the durations of all contingent links that finish prior to a timepoint x. (This may be viewed as a partial mapping from C to $\mathbb{R}$.) We call this the *partial execution* of T with respect to x, denoted by $T_{\prec x}$.

Then an *execution strategy* S is a mapping $S : \mathcal{P} \rightarrow \mathcal{T}$ where $\mathcal{P}$ is the set of projections and $\mathcal{T}$ is the set of schedules. An execution strategy S is *viable* if $S(p)$ is consistent for each projection p.

We are now ready to recall the various types of controllability [12, 7].

An STNU is *Weakly Controllable* if there is a viable execution strategy. This is equivalent to saying that every projection is consistent.

An STNU is *Strongly Controllable* if there is a viable execution strategy S such that

$$[S(p1)]_x = [S(p2)]_x$$

for each executable timepoint x and each projections $p1$ and $p2$. Thus, a Strong execution strategy assigns a fixed time to each executable timepoint irrespective of the outcomes of the contingent links.

An STNU is *Dynamically Controllable* if there is a viable execution strategy S such that

$$[S(p1)]_{\prec x} = [S(p2)]_{\prec x} \Rightarrow [S(p1)]_x = [S(p2)]_x$$

for each executable timepoint x and each projections $p1$ and $p2$. Thus, a Dynamic execution strategy assigns a time to each executable timepoint that may depend on the outcomes of contingent links in the past, but not on those in the future (or present). This corresponds to requiring that only information available from observation may be used in determining the schedule.

It is easy to see from the definitions that Strong Controllability implies Dynamic Controllability, which in turn implies Weak Controllability. Strong Controllability is known to be tractable and Weak Controllability is known to be co-NP-complete [13, 5]. Dynamic Controllability was expected to be hard to check, primarily because of a time asymmetry where a control decision may depend on the past but not on the future, but a recent work [7] demonstrated that this was actually a tractable problem.

We designed a local path consistency -like process, that is an algorithm analysing every triangle of links, restricting the bounds on a link according to the two other links in the triangle. Which required to distinguish between several cases according to the type of the links in the triangle (contingent or requirement). A first sound but incomplete algorithm was designed in such a way and called 3DC. Deepening the analysis a bit, we discovered that in some cases new types of constraints were needed to meet the dynamic execution strategy needs. Those were called *wait* constraints : a wait $< C, t >$ on the link from time-point A to time-point B means *"once A is released, the agent must wait for the possible occurrence of C at least t time units before she might release B"*. This is often required to ensure the execution will be safe whatever the precise time of occurrence of the (observed) event C is. We proved that such *wait* constraints could be easily added to the STNU model and *regressed*, which is a backward propagation process, still made locally through triangles, and thus still tractable. The resulting enhanced algorithm is called 3DC+, is sound and complete with respect to global Dynamic controllability checking, and provides the minimal network (i.e. in which all inconsistent values have been removed). All details on 3DC and 3DC+ are out of the scope of this paper and can be found in [7].

3 Task Sequencing in Scheduling

3.1 Scheduling with resource and time constraints

Scheduling [10] is the problem of assigning resources and setting start times to a set of given tasks. The constraints to be satisfied are both in terms of resource (type of resource demanded by each task) and time (partial order of tasks corresponding to a given process to be carried out). For example, the well-known Job-Shop Scheduling problem [2, 3] addresses jobs, i.e. sequences of tasks (e.g. turning, drilling, etc) complying with a given routing, that need to be carried out simultaneously in a manufacturing process. Resources are machines that are dedicated to certain tasks but might only process one task at a time (i.e. their capacity equals to one). Those are called *disjunctive* resources, since two tasks competing on the same resource will have to be *sequenced*. We will call any such pair a *critical* pair of disjunctive tasks, and we will use letters i and j to refer to them, i^- [resp. i^+] being the start [resp. end] time-point of task i, and similarly for j. The disjunctive constraint is hence of type i *before or after* j.

If each task might choose among alternative resources, then the system will actually have to carry out three steps : (1) *allocate* a resource to each task, (2) *sequence* the sets of tasks requiring the same resource, and (3) *schedule* each task by setting its start time.

Of course, more complex types of resource might be considered, such as cumulative or reservoir constraints [4], but we will not address them in this paper. Moreover, we will as a first step only tackle the sequencing problem, which means we will consider that resources are already allocated to the tasks.

Scheduling is not only addressed in the manufacturing community as a stand-alone process. It also appears as a sub-process of planning : once the tasks to reach a given goal have been selected, resource allocation, task sequencing and precise timing must also be conducted (see [10] for a comprehensive survey).

3.2 Sequencing rules and temporal constraint networks

As said before, a number of planning and scheduling systems [10, 8, 6] rely on temporal constraint networks. One may choose between the general TCN framework where disjunctions of intervals of duration are allowed but in which propagation techniques are either exponential or incomplete [9], and the less expressive STN formalism, that provides polynomial time complete temporal consistency checking. The latter choice is ours, especially as extending to STNU preserves this tractability result.

Anyway TCN and STN are binary constraint models. Therefore none of them might express a sequencing constraint i *before or after* j, which indeed amounts to the 4-ary constraint between time-points $i^+ \prec j^-$ *or* $j^+ \prec i^-$.

This is a well-known result in qualitative time reasoning, time-point algebra [14] being less expressive but more efficient than interval algebra [1] : the latter indeed makes it possible to directly express the constraint *before or after* as a binary constraint between two interval variables. Not only are interval algebra NP-complete with respect to temporal consistency checking, but they also make it difficult to represent durations and dates of instantaneous events.

As far as time-point models are considered, one has two possibilities. First, one may rely on extended models allowing non-binary constraints between time-points, in the shape of disjuntions of binary link constraints, namely the Disjunctive Temporal Problems (DTP) [11]. But that means applying intractable general solving methods.

The second choice is to reason over two distinct stages, first tackling the sequencing problem, enforcing for each pair of disjunctive tasks one of the two sequences, then checking temporal consistency in the resulting TCN [resp. STN]. This is what is done in [2] where sequencing decisions are looked for in a search tree, checking consistency at each step in the partially sequenced STN-like model (extended to account for fuzzy durations), backtracking in case of failure. This process will of course be exponential in the worst case, but the complexity is confined to the pairs of disjunctive tasks.

A similar process is suggested in [3], in which the three steps, allocation, sequencing and scheduling are addressed, and the duration of a task depends on the allocated resource but is reduced to a single value. We will call $d_{i,k}$ the fixed duration for task i when assigned to resource k. Domains of time-points x are represented as intervals of possible values $[l(x), u(x)]$, and duration of tasks as disjunctions of single-value intervals, hence the resulting temporal model amounts to a kind of general TCN. Instead

of searching through a tree, filtering techniques are applied to the sequencing problem, thanks to a set of rules inspired by the Operations Research community. These are alternated with incomplete temporal propagations in the TCN-like model. For instance the *Forbidden Sequence (FS)* rule checks if one of the two choices is inconsistent with the domains of the time-points : if i and j compete for the resource k, this gives

$$\texttt{if } u(j^+) - l(i^-) < d_{i,k} + d_{j,k} \qquad \texttt{then } NOT(i\ before\ j)$$

Then the choice j *before* i is compulsory, the link $j^+ \prec i^-$ is added and propagated in the TCN. Such propagations might in turn remove possible duration values, which means removing possible allocation choices.

More elaborated rules exist (see [3] for details), for instance to check that a task cannot be inserted in a set of disjunctive tasks, but they induce computations of higher complexity, therefore we decided to ignore them in this paper. On the contrary, we will manage to adapt the FS rule to our context, namely in the context of STNU, disregarding for now the allocation problem but using all the information directly available in the minimal STNU obtained after applying 3DC or 3DC+. But before describing our technique, we first present a formal analysis of the so-called *Sequencing-ability* properties in a STNU.

4 Sequencing-abilities in STN with Uncertainty

Just like controllability, sequencing-ability is a property of a temporal plan. Intuitively, it is the possibility or not to sequence any pair of tasks competing on the same resource. Which means one needs to model both the STNU and the sets of disjunctive tasks.

Sequencing-ability is highly connected to controllability : the possibility to sequence means that the execution strategy is still viable when it also considers sequencing constraints. But this might be viewed in two distinct ways.

The first one is to add the sequencing constraints in the temporal model. Then these constraints must be satisfied as any other one. A viable strategy S will be as before a mapping $S : \mathcal{P} \rightarrow \mathcal{T}$ from the set of projections to the set of schedules, and it will be viable if $S(p)$ is consistent for each projection p. Consistency means here the satisfiability of both usual temporal constraints and sequencing ones. Then the three levels of controllability can be defined as before. The first problem is that, as said before, sequencing constraints cannot be directly expressed in a STN. Which means one should analyse controllability in an extended model, namely a DTP distinguishing between requirement and contingent links, in other words a DTPU (Disjunctive Temporal Problem with Uncertainty). One would hence lose the effectiveness of controllability checking in STN. The second problem is that all constraints are considered in the same way, and hence the same level of controllability is enforced. We would rather let the user of the system tune such levels independently : for instance enforcing static sequencing (made off line, before execution), but dynamically scheduling the tasks, should remain possible. Therefore, we chose to keep relying on the STN formalism, considering three levels of sequencing-ability that are additional demands on the execution strategy S, such as the three levels of controllability are.

First we need to define the critical pairs, in which each task may be either a requirement or a contingent link. An *allocation* A is a mapping $A : E \rightarrow K$ where E is the set of links and K the set of disjunctive resources. A virtual resource called `none` is the default result, meaning this link does not correspond to any task needing a resource. Then the function `critical` takes as input two links i and j and returns `True` if and only if $A(i) = A(j) \neq$ `none` .

Now we can define the three levels of sequencing-ability :

- *Weak Sequencing-ability* (or *WS-ability* for short) of an STNU is met if and only if, for all pairs (i, j) such that `critical`(i, j), for all projection p,

$$[S(p)]_{i^+} < [S(p)]_{j^-} \quad or \quad [S(p)]_{j^+} < [S(p)]_{i^-}$$

- *Strong Sequencing-ability* (or *SS-ability* for short) of an STNU is met if and only if it is WS-able and if, for all pairs (i, j) such that `critical`(i, j), for all pairs of projections (p_1, p_2),

$$[S(p_1)]_{i^+} < [S(p_1)]_{j^-} \quad \Rightarrow \quad [S(p_2)]_{i^+} < [S(p_2)]_{j^-}$$

  which means the chosen sequence is the same in all situations.

- *Dynamic Sequencing-ability* (or *DS-ability* for short) of an STNU is met if and only if it is WS-able and if, for all pairs (i, j) such that `critical`(i, j), for all pairs of projections (p_1, p_2),

$$[S(p_1)]_{\prec ij} = [S(p_2)]_{\prec ij}$$

$$\Rightarrow \quad (\quad [S(p_1)]_{i^+} < [S(p_1)]_{j^-} \Rightarrow [S(p_2)]_{i^+} < [S(p_2)]_{j^-} \quad)$$

  where $[S(p)]_{\prec ij}$ is the partial execution of $[S(p)]$ with respect to both the start times of i and j, which means the sequencing may depend on the outcomes of contingent links in the past, but not on those in the future (or present).

Here again it is easy to see that SS-ability implies DS-ability which in turn implies WS-ability. Then one might consider all possible combinations of controllability and sequencing-ability demands. First if the STNU must be Strongly controllable, then all start times of tasks will be set statically, which means if there is a sequencing, it will necessarily be the same in all situations. In other words, only SS-ability is meaningful, and actually all three levels of sequencing-ability are equivalent : WS-ability checking is enough to ensure SS-ability.

Conversely, if the STNU is asked to be merely Weakly controllable, then the three levels of sequencing-ability might theoretically apply. But from a practical point of view, Weak controllability is useful in situations in which the whole situation is known just before the execution begins [13]. Then, WS-ability is enough, and there is no reason to try to enforce an upper level.

The most interesting case is when one wants a Dynamically controllable STNU. First, SS-ability might be relevant. This is actually what we have considered until now,

that is a *static* sequencing processed off line, which is unique, followed by a dynamic scheduling processed on line.

Then, one would guess that as above, since the execution strategic is dynamic, the sequencing choices will necessarily be dynamic, and therefore WS-ability and DS-ability are equivalent, the former being sufficient to enforce the latter. But things are a little more subtle, as we are going to show. Let us try to prove it *ad absurdum*. Suppose the STNU is Dynamically controllable and WS-able, and suppose the DS-ability is not met, i.e. there exist i, j, p_1, p_2, with at least one of i and j being a contingent link, such that

$$[S(p_1)]_{\prec ij} = [S(p_2)]_{\prec ij}, \quad [S(p_1)]_{i^+} < [S(p_1)]_{j^-}, \quad [S(p_2)]_{i^+} > [S(p_2)]_{j^-}$$

In other words sequencing choices are different in p_1 and p_2. If ever the durations of i and j are the same in p_1 and p_2, then p_1 and p_2 are equal up to and including i and j, while the decisions taken for timing i and j differ, which contradicts the Dynamic controllability assumption. So the only possibility is that the durations of i and j are distinct in p_1 and p_2, and this difference is the reason why i was sequenced before j in one case and j before i in the other one. But this can be overome very easily.

Locally, the sequencing decision should be taken (dynamically) before one starts either i or j. Therefore, if at least i or j is a contingent link, in a dynamic execution strategy such a decision should be made irrespective of the effective durations of i and/or j. This is a local DS-ability property :

- Local DS-ability is enforced in a STNU if and only if, for any critical pair (i, j), at least one of the two sequencing choices is possible for all possible durations of the contingent links i and/or j

Then the following property holds : DS-ability is enforced in a STNU if and only if both WS-ability and local DS-ability are enforced.

In the next sextion we will give an algorithm for checking local DS-ability in polynomial time, which is actually an adaptation of the Forbidden Sequence rule to the STNU context.

5 A Filtering Technique for local DS-ability checking

To enforce local DS-ability, it is enough to build a filtering algorithm that will check any critical pair in the STNU, which means check 4-uples of time-points. This is why we will call such an algorithm 4DS[1].

i and j are the involved links corresponding to the tasks. Since the 3DC and 3DC+ algorithms for checking Dynamic controllability compute the minimal network, all other links in this local network of 4 time-points are present with corresponding lower and upper bounds. We might hence consider all the following intervals of durations : $[l(i), u(i)]$ and $[l(j), u(j)]$ of course, but also $[l(i^-j^-), u(i^-j^-)]$, $[l(i^-j^+), u(i^-j^+)]$, $[l(i^+j^-), u(i^+j^-)]$ and $[l(i^+j^+), u(i^+j^+)]$. Obviously $l(i^+j^-) < 0$ and $u(i^+j^-) > 0$:

[1] Please note nevertheless that not all 4-uples of variables are checked, so this algorithm is more restricted than a general scope 4-consistency filtering technique.

if not, then one would already have i and j sequenced, hence they would not form a critical pair. The same arises for $l(i^-j^+)$ and $u(i^-j^+)$. We will not consider these links in the remainder, but only i^-j^- and i^+j^+.

We will also denote $d(i)$ [resp. $d(j)$] as the duration of i [resp. j] to consider according to the type of link : $d(i) = l(i)$ when i is a requirement link (one will consider best cases) and $d(i) = u(i)$ when i is a contingent link (one will consider worst cases). Then the following rules hold.

- If $l(i^-j^-) > 0$ and/or $l(i^+j^+) > 0$ then j can never be executed before i : $i^+ \prec j^-$ is added and propagated. The converse case is when $u(i^-j^-) < 0$ and/or $u(i^+j^+) < 0$.
- Else, if $u(i^-j^-) < d(i)$ and/or $u(i^+j^+) < d(j)$ then in case of contingent links that means there is at least one situation in which j cannot be executed before i : $i^+ \prec j^-$ is added and propagated. The converse case is when $-l(i^-j^-) < d(j)$ and/or $-l(i^+j^+) < d(i)$.
- In all remaining cases, both sequencing choices are *a priori* still possible in all situations, hence nothing is done.

One can see this process is very similar to the FS rule, but it needs to take into account the type of links : in case of requirement links, the sequencing is forbidden if as usual it is strictly not possible because not enough slask is available between the tasks, while in case of contingent links, it is forbidden if it is not possible in at least one situation. Moreover, our rules extend the FS rule in that the minimal network provide more information in terms of delays between end points of i and j, which is more precise than just the domains of these time-points.

Then, the system will behave as in [3], alternating temporal propagation and local DS-ability checking. Of course, if both sequencing choices are forbidden, the algorithm returns a global inconsistency. If a sequencing must be enforced, that becomes a static sequencing, which will of course be propagated through 3DC or 3DC+, but will also reduce the number of dynamic sequencing choices that are left to be made on line.

6 Conclusion and Future Work

Following previous work on dynamic properties of temporal networks in the presence of uncertainties on duration constraints, this paper has given a thorough characterization of the DS-ability, which ensures that dynamic sequencing will also be made possible during execution without any constraint violation.

The proposed 4DS algorithm checks this property locally, for any pair of disjuntive tasks. It is a sound but incomplete process that ideally complements 3DC+ to deal with both dynamic sequencing and dynamic scheduling abilities. It will help filtering out inconsistent sequencing choices, but will also facilitate a global DS-ability checking process that still needs to be designed. We have proven that it now merely remains to prove WS-ability, which is going to be the next step of our research work.

A number of extensions of 4DC are possible. For instance, we have not taken into account the *wait* constraints that are added by 3DC+. They imply a few more rules that

can be easily incorporated in the process. Another line of research is to compare our approach to more systematic ones like for instance using DTPU, that is DTP extended to account for Uncertainty, as in STNU. Last but not least, the work done suggests the possibility to enlarge the idea to get the whole picture, resource allocation being also made on line in a dynamic fashion. It seems that the allocation rules given in [3] might very well be adapted to the STNU framework.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):509–521, 1983.

[2] D. Dubois, H. Fargier, and H. Prade. The use of fuzzy constraints in job-shop scheduling. In *IJCAI-93 Workshop on Knowledge-Based Planning, Scheduling and Control*, Chambéry (France), 1993.

[3] M-J. Huguet and P. Lopez. An integrated constraint-based model for task scheduling and resource assignment. In *CP-AI-OR'99, 1st Int'l Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems, Ferrara (Italy)*, 1999.

[4] P. Laborie. New algorithms for propagating resource constraints in ai planning and scheduling. In *Working notes of the IJCAI'01 workshop on Planning with Resources*, Seattle (USA), 2001.

[5] P. Morris and N. Muscettola. Managing temporal uncertainty through waypoint controllability. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on A.I. (IJCAI-99)*, pages 1253–1258, Stockholm (Sweden), 1999. Morgan Kaufmann.

[6] P. Morris, N. Muscettola, and I. Tsamardinos. Reformulating temporal plans for efficient execution. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, Trento (Italy), 1998.

[7] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *(soumis à) International Joint Conference on A.I. (IJCAI-01)*, Seattle (WA, USA), 2001.

[8] P.Laborie and M. Ghallab. Planning with sharable constraints. In *Proceedings of the 14th International Joint Conference on A.I. (IJCAI-95)*, Montreal (Canada), 1995.

[9] E. Schwalb and R Dechter. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61, 1997.

[10] D.E. Smith, J. Frank, and A.K. Jónsson. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1), 2000.

[11] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120:81–117, 2000.

[12] T. Vidal. Controllability characterization and checking in contingent temporal constraint networks. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge (Co, USA), 2000. Morgan Kaufmann, San Francisco, CA.

[13] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11:23–45, 1999.

[14] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms: a revised report. In *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufman, 1989.